University of Crete
School of Sciences and Engineering
Computer Science Department

Towards a universal Service Network-centric framework to design, implement and monitor Services in complex Service Ecosystems: The Service Network Analysis & Prediction Tool (SNAPT)

by Pantelis Petridis

Master's Thesis

Heraklion, September 2010

University of Crete
School of Sciences and Engineering
Computer Science Department

Towards a universal Service Network-centric framework to design, implement and monitor Services in complex Service Ecosystems: The Service Network Analysis & Prediction Tool (SNAPT)

by Pantelis Petridis

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

Author: _____
Pantelis Petridis, Computer Science Department

Board of enquiry:

Supervisor: _____
Christos Nikolaou, Professor - University of Crete

Member: _____
Dimitrios Plexousakis, Professor – University of Crete

Member: _____
Kostantinos Maggoutis, Researcher – FORTH-ICS

Approved by: _____
Panos Trahanias, Professor – University of Crete, Chairman of the Graduate Studies Committee

Towards a universal Service Network-centric framework to design, implement and monitor Services in complex Service Ecosystems: The Service Network Analysis & Prediction Tool (SNAPT)

Pantelis Petridis

Master's thesis

University of Crete – School of Sciences and Engineering
Computer Science Department

Abstract

As Service-Oriented Architecture becomes the industry standard to deliver complex, reusable and reliable services, while at the same time traditional economic and strategy analysis methodologies are still used to achieve business performance, the need to smoothly make a transition from the early service design stages to service implementation (and vice-versa) is critical. Service Networks offer the required level of abstraction for this effort to yield fruit, as they study both existing and emerging business relations between service providers and service consumers. We aim at developing a Service Network-centric framework for modeling, studying, developing and monitoring services in complex Service Ecosystems, where all involved stakeholders can provide and extract information to facilitate their needs. In this work we propose the Service Network meta-model, that identifies all entities participating in a Service Network and imprints their strategic goals and requirements. We also present a simple cost - revenue model to enable Service Network calculations. To support our methodology we have developed the Service Network Analysis & Prediction Tool (SNAPT). SNAPT follows a flexible Model-Driven Architecture approach to allow for the integration with future extended Service Network models. As a result, SNAPT is designed to serve as an intermediate Service System design and analysis platform. To demonstrate the application of Service Network analysis in Service Ecosystems, we also present an iterative process that exploits Service Networks to enable better service selection in the automotive industry domain using User Requirements techniques.

Προς ένα πλαίσιο ενιαίο και βασισμένο στα Δίκτυα Υπηρεσιών, για σχεδίαση, υλοποίηση και εποπτεία υπηρεσιών σε σύνθετα οικοσυστήματα: Το εργαλείο σχεδίασης και ανάλυσης Δικτύων Υπηρεσιών SNAPT

Παντελής Πετρίδης

Μεταπτυχιακή εργασία

Πανεπιστήμιο Κρήτης - Σχολή Θετικών & Τεχνολογικών Επιστημών
Τμήμα Επιστήμης Υπολογιστών

Περίληψη

Τα τελευταία χρόνια, ολοένα και περισσότεροι οργανισμοί και επιχειρήσεις χρησιμοποιούν αρχιτεκτονικές συστημάτων που προσανατολίζονται στις υπηρεσίες (Service-Oriented Architectures - SOA) προκειμένου να παρέχουν σύνθετες, επαναχρισημιοποιήσιμες και αξιόπιστες εφαρμογές. Ταυτόχρονα, παραδοσιακές οικονομικές θεωρίες και μεθοδολογίες χρησιμοποιούνται για τη μελέτη συστημάτων υπηρεσιών και την παρακολούθηση της απόδοσης των επιχειρηματικών διαδικασιών. Επομένως, η ανάγκη για τη δημιουργία μιας ενιαίας μεθοδολογίας που θα καλύπτει όλο το φάσμα των διαδικασιών από τη σχεδίαση έως την υλοποίηση και εποπτεία συστημάτων υπηρεσιών, είναι επιτακτική. Τα Δίκτυα Υπηρεσιών (Service Networks) εξυπηρετούν τη μελέτη των σχέσεων που αναπτύσσονται ανάμεσα στους παροχείς και τους καταναλωτές υπηρεσιών, και επομένως μπορούν να αποτελέσουν ένα κοινό σημείο επαφής ανάμεσα στους σχεδιαστές και αναλυτές υπηρεσιών και τους μηχανικούς λογισμικού και να συντελέσουν τον πυρήνα μιας τέτοιας μεθοδολογίας. Σε αυτή την εργασία, παρουσιάζουμε ένα μοντέλο Δικτύων Υπηρεσιών, που αναγνωρίζει όλες τις οντότητες που συμμετέχουν σε ένα τέτοιο δίκτυο και αποτυπώνει τις στρατηγικές τους. Επίσης, προτείνουμε ένα απλό μοντέλο για τον υπολογισμό του κόστους και του κέρδους των συμμετεχόντων σε ένα Δίκτυο Υπηρεσιών. Για τους σκοπούς της μεθοδολογίας μας, υλοποιήσαμε ένα εργαλείο σχεδίασης και ανάλυσης Δικτύων Υπηρεσιών με το όνομα SNAPT (Service Network Analysis & Prediction Tool). Το εργαλείο ακολουθεί μια ευέλικτη «μοντελο-κεντρική» αρχιτεκτονική (Model Driven Architecture – MDA) προκειμένου να μπορεί να ενσωματώσει στο μέλλον πιθανές επεκτάσεις του μοντέλου Δικτύων Υπηρεσιών. Επιπλέον, για να αναδείξουμε τις εφαρμογές της ανάλυσης Δικτύων Υπηρεσιών σε οικοσυστήματα υπηρεσιών, παρουσιάζουμε ένα παράδειγμα από την αυτοκινητοβιομηχνία, οπου τα Δίκτυα Υπηρεσιών χρησιμοποιούνται για να βελτιστοποιήσουν την διαδικασία επιλογής υπηρεσιών με χρήση τεχνικών από τον τομέα της ανάλυσης απαιτήσεων (Service Requirements Engineering).

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my supervisor, Professor Christos Nikolaou for his guidance and support during this thesis and for giving me the opportunity to follow the postgraduate program of the Computer Science Department and be a member of the Transformation Service Laboratory (TSL) team. I would also like to express my thanks to Professor Dimitrios Plexousakis and Kostas Maggoutis for being part of my board of enquiry and for their wholesome comments and feedback.

I also wish to especially thank my friend and colleague Giorgos Stratakis. This work would never have been possible without his indispensable help, advice and support.

Special thanks to my close friends for their love and patience over the past few years and for helping me pull through hard times. Thank you all.

I dedicate this work to my family. Their love and belief in me have been the driving force in my life.

1. Introduction

As web services technologies mature, and first generation service-oriented architectures are widely adopted, a new revolution of service orientation is emerging – the procurement of web services into different markets [1]. While web services are exposed and connected with one another, they give rise to service ecosystems – a logical collection of services whose exposure and access are subject to constraints [1]. The underlying business models of current service ecosystems are mostly about bringing service consumers closer to service providers, or allowing services to be accessed through service brokers with collected revenue then passed back to providers. A service provider is able to deploy and deliver endpoints to different business channels without having to factor in service delivery functionality.

Service Networks can be seen as a currently prominent example of a service ecosystem. Service Networks have been proposed to analyze and optimize company's business collaborations by modeling their interactions using networks [2]. A Service Network is a graph-based approach to model a business environment as a set of business partners and their relations. Service Networks reside on a high abstraction business level depicting partners as nodes and their offering and revenues as edges.

An ICT-enabled service ecosystem provides intermediation mechanisms for service providers and consumers to meet and interact, such as brokers, registries, search and discovery services, monitoring, reputation and recommendation services, legal enforcement and compliance services, etc. Initially, an observer outside the ecosystem would observe an almost chaotic, random and continuous creation and destruction of service provider and service consumer links; we call them dipoles. Examples of these dipoles are abundant in today's Internet: Amazon and EBay provide services to human consumers (buy books, match buyers with sellers, etc.), web service mash-ups, etc. Interactions (messages and data exchanged, sequencing of messages, etc.) are usually called *business processes* [3] and are usually governed by accepted or proposed standards (e.g. RosettaNet standards [4], industry sector standards such as Open Travel [5], etc.).

As time progresses, some links occur more frequently than others, indicating a preference of the consumers for some of the services offered. Patterns of interaction

thus occur, as some links (dipoles) are almost always active between a service provider and a service consumer. This enduring over time service dipole is the simplest form of a Service Network. More complex Service Networks are formed when the service provider composes services from possibly several other service providers to offer his own service; for example, a travel agent combines services provided by airline companies, hotels, rental cars, sightseeing tour operators, museums, etc. to create a travel package offering for her customers. Airline companies in turn may combine flights from regional airlines with which they are cooperating (for example through code sharing). Business processes are now spanning several service providers with end-to-end business transactions (for example, for the travel package to be finalized hotels will have to reserve rooms, airlines to reserve flights, etc.).

Within a service ecosystem there may exist several (possibly competing) Service Networks, where certain service providers work together in order to deliver a value creating service, which is composed out of services offered by individual service providers. A Service Network is based on strong, long-term relationships where network participants trust each other. Obviously, each partner of a Service Network has her own business and performance objectives that she tries to achieve. These objectives are usually codified through the Key Performance Indicators (KPIs) that set measurable goals to be achieved in a certain time frame [6]. Typical KPIs refer to delays, cost, revenues, profits, usefulness usually defined through some utility function, etc. [7].

Since Service Network theory is a natural evolution of Michael Porter's value chain concept [8][9], several approaches have been proposed for studying and analyzing Service Networks – referred to as Service Value Networks or simply Value Networks – from the Economic Sciences viewpoint. As a result, there exists a gap between the economic and ICT perspectives on Service Networks. In this thesis, we propose a Service Network model taking into account related work performed by researchers such as Marina Bitsaki et al. [10], Verna Allee [11], Jaap Gordijn [12] etc. We also incorporate the KPI concept in order to reflect business goals on the Service Network level. At the same time, we stay focused to the goal of providing a common language that both economists and ICT developers will be able to understand: The proposed model aims to act as a hub between traditional Value Network analysis and service-oriented ICT systems and models.

To support our methodology, much effort has been put into delivering a tool that will enable Service Network modeling and analysis; we name it *Service Network Analysis & Prediction Tool* (SNAPT). The tool targets business analysts who need to study existing Service Networks or explore the vitality of emerging Service Networks. In addition to

modeling and studying Service Networks, SNAPT users are also allowed to develop their own KPI Library and map business goals to the Service Network model. Although beyond the scope of this work, supplementary implemented functionality such as the conversion to Abstract Business Processes, the report generation mechanism adopting methodologies such as Verna Allee's Value Network Analysis (see section 2.2.1) etc, elevates SNAPT to a standalone Service System designing platform.

Demonstrating the application of Service Networks to Service Systems in practice, has been another task of this thesis. We have developed a process that utilizes Service Network analysis in conjunction with established requirements engineering techniques in order to improve service discovery and selection within a Service Ecosystem. We also use a real-world scenario based on the automotive industry domain to exhibit our process and discuss our findings.

The remainder of this work is organized as follows: In the next section we present background theory related to Service Ecosystems, Service (Value) Networks and Key Performance Indicators. In chapter 3, the proposed Service Network model is discussed. We present a Service Network meta-model, exhibit our notation for Service Networks and explain the Cost-Revenue model used in the service value estimation process. The application of Service Network analysis to enhance service selection within a service system is discussed in chapter 4. Chapter 5 introduces the "Service Network Analysis & Prediction Tool". We present the technologies used to author the tool, expand on the concept of model-driven software development and overview SNAPT's architecture. Particular issues concerning the SNAPT development process as well as selective design decisions are later on presented in chapter 6. Finally, the concluding section can be found in chapter 7. After summarizing the work performed in the context of this thesis, we dive into future challenges that are yet to be addressed.

2. Background

2.1. Service Systems & Service Ecosystems

Over the past two decades, firms and industries tend to transform from a goods orientation to a service orientation [13]. These radical changes have been mainly motivated by the general transformation in developed countries from industrial economies to service-led economies. In order to understand services and facilitate the needs of a service-based business world, new concepts and practices should be exposed. One of the most fundamental and wide-spread terms describing such systems is the concept of *Service Systems*.

In [14] Service Systems are defined as "…dynamic configurations of resources (people, technology, organizations and shared information) that can create and deliver service while balancing risk-taking and value co-creation". Karni and Kaner argue that systems in general are identified by nine key elements (classes): Customers, goals, inputs, outputs, processes, human enablers, physical enablers, informatics enablers and environment. They state that Service Systems should be distinguished from other systems in the sense that an end customer could be involved in all nine classes of the Service System [15]. According to Jim Spohrer, Service Systems should be capable of improving other systems through applying their resources –i.e. the "other" system gains value -. At the same time, Service Systems should also be capable of improving their own state by getting access to resources from other systems (i.e. the system itself gains value from interactions with others). In this context, economic transactions depend on value creation between Service Systems, where each system must willingly interact and both systems must be improved [16]. In this context, Spohrer perceives value to be an improvement for the Service System measured by the system itself. The environments in which Service Systems can co-exist and interact with each other are called *Service Ecosystems*.

Service Ecosystems form "market places for trading services in the business sense and involve actors from different legal bodies" as stated in [17]. Service Ecosystems are tightly coupled with the evolution of the Internet and the rise of Service-Oriented Architectures. Online marketplaces on the Web, where providers can propose and clients discover and select services while being able to negotiate and agree on service

levels, are called Web Service Ecosystems [18]. Web Service Ecosystems -contrary to application servers- are composed of services that are potentially outsourced in the ecosystem –i.e. several alternative web payment services could be available within a Web Service Ecosystem and participants could incorporate any of these payment engines to already running service instances-. Service delivery can be constrained depending on the ecosystem's underlying business model; this is also unlikely as far as application servers are concerned. Client constraints regarding service discovery, ranking, payment, monitoring, etc. can be contractual. From the supplier side, regulations on how services are published, composed, brokered or licensed can be provided [19]. Web Service Ecosystems rely on one or several brokers who create value by either connecting service providers with service consumers or by enabling consumers to access services providing intermediary functionality such as payment; in this work, we call such services "Enablement Services" and their providers "Enablers" (see section 3.1). Of course, there is no rule preventing any actor to play multiple roles within a Web Service Ecosystem. A representation of the top-level architecture of a Web Service Ecosystem can be found in figure 1.

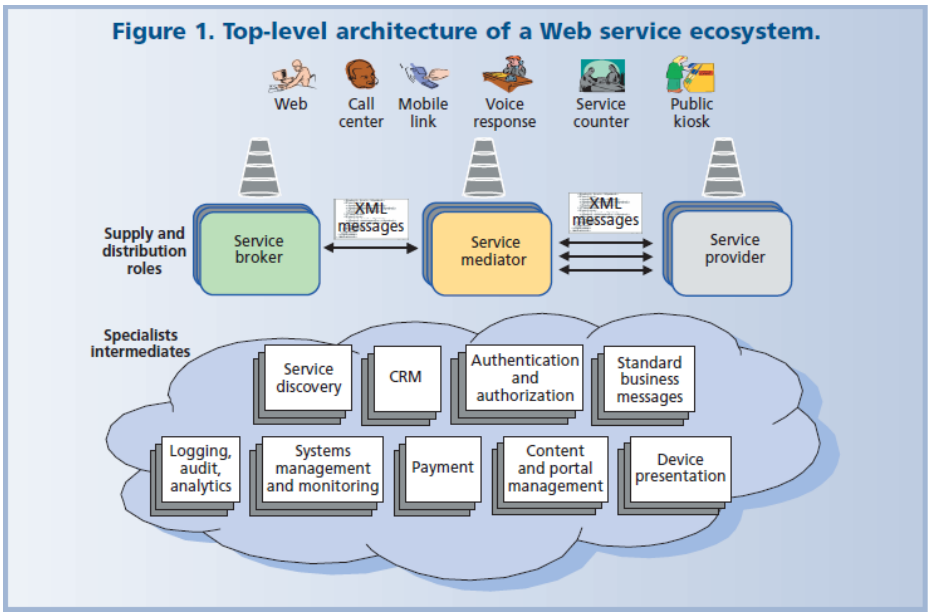

Figure 1. The Web Service Ecosystem. Source [ref]

In order to study and analyze Services, Service Systems and Ecosystems while being able to identify their origins and predict their evolution a new Science of Services or Service Systems has been proposed [14]. In [20], it is argued that such Science could benefit from a consistent and inclusive set of ideas that would help in interpreting service research and practice and in organizing instructional programs; the fundamentals of Service Systems might meet this need as all significant services are delivered through service systems.

2.2. Service (Value) Networks

As discussed in the introductory section, Service Ecosystems evolve and interactions between service providers and consumers constantly occur, networks between certain providers, services and consumers are formed; we call such networks Service Networks. In this section, we will focus on research conducted on the field of Service Networks and expand on relevant methodologies. Some of the presented approaches apply the most to economists and business managers while others consider Service Networks from an ICT perspective. As a result, some researchers define and interpret Service (Value) Networks in such a way that makes it impossible to converge with others. Note also that some approaches refer to Service Networks as "Service Value Networks" or just "Value Networks". Finally, some methodologies adopt declining views on value definition as well. Such differences are also noted in the next sections.

2.2.1. The ValueNet Works methodology

First, we will discuss the "*ValueNet Works*" methodology introduced by Verna Allee[1]. Allee identifies three key elements within a Value Network: Participants, Transactions and Deliverables. Participants can be individuals or groups of people carrying out several activities. Transactions occur in the context of the activities performed by the Participants. Finally, deliverables represent value objects that the Participants exchange. Allee classifies deliverables to *tangibles* and *intangibles*, where tangibles involve products or services generating revenues and intangibles are non-contractual services – i.e. knowledge sharing, spread of information etc- [21]. Consequently, Allee proceeds by defining Value Networks as "…any web of relationships that generates both tangible and intangible value through complex dynamic exchanges between two or more individuals, groups or organizations." [22]. According to Allee, the first task to be performed when modeling a Value Network is to identify all tangible and intangible exchanges occurring in the network. The Value Network graph is then constructed by modeling the Roles that Participants play as nodes, and representing all transactions as arrows following the direction of the deliverables. Allee uses color-coding to identify whether a transaction is tangible or intangible. As soon as the Value Network is modeled, one can proceed with the Value Network analysis. The analysis is performed by means of calculated indices that describe the Value Network. Core elements of the proposed methodology are the "exchange analysis", the "impact analysis" and the "value creation" analysis [11]. *Exchange analysis* is performed to identify the network's exchange patterns. The value flow is examined to determine whether transactions –tangible and intangible- are

[1] http://www.vernaallee.com/

"healthy" –i.e. a role is benefiting against another role etc-. As soon as exchange analysis is completed all network's imbalances and bottlenecks should be exposed. Key indices of the exchange analysis are –among others- the ratio of tangible to intangible transactions, the number of tangible and intangible transactions that each role performs etc. The *impact analysis* focuses on determining whether roles can convert their inputs to beneficial assets that will enable their survival in a competitive market. Assets are divided to financial, business relationships, competence and structure. Impact analysis reveals if a transaction comes to a cost or benefit and how the recipient perceives the incoming value. Typical impact analysis indices include the perceived value (receiver) per asset type, the asset impact of intangible and tangible transactions and the distribution of asset impact by role. Last but not least, the *value creation analysis* also deals with cost / beneficial transactions but this time focusing on a role's outputs. Indices of the value creation analysis include the number of outgoing connections for both tangible and intangible transactions, the percentage of deliverables generated by each role, the perceived value (sender) per asset type etc. To supplement the ValueNet Works methodology Verna Allee has developed the "*Insights"* web-based tool [23], the successor to an older tool called "GenIsis". The user describes Value Networks using Excel spreadsheet templates, that are later on uploaded to the tool. Then the ValueNet Works analysis is performed and the tool generates reports that not only visually represent the Value Network, but also calculate all the indices included in the ValueNet Works methodology. As with GenIsis, the Insights tool is commercial and proprietary. Although not covered in this thesis, the Service Networks Analysis & Prediction Tool that we have developed does incorporate the ValueNet Works methodology in order to provide a sophisticated qualitative analysis of Service Networks.

2.2.2. The e3value methodology

The e3value methodology, proposed by Jaap Gordijn is another interesting approach to Service Networks. The entities comprising an e3value model are Actors, Market Segments, Value Objects, Value Ports, Value Interfaces, Value Exchanges, Value Offerings and Value itself [24]. *Actors* are the equivalent to what we have called "Participants" or "Business Entities" so far: economic entities that perform several activities in exchange to revenues. In this model, classes of actors that appear to have similar characteristics are grouped in *market segments*. *Value* refers to the profitable activities that actors perform within a Value Network. *Value ports*, grouped into *value interfaces*, are the actors' input and output ports enabling the exchange of *value objects*. Value objects are contained in *value exchanges* which are grouped to *value offerings*. The e3value model is supplemented by Use Case Maps (UCM) to provide additional process sequencing

information -called *scenario paths*- that indicate the way value objects flow within the network. To follow the e3value methodology [25], one has to first identify an end customer need – the "final" service that the value network will deliver- and decompose it to services that are already known to be offered by several service providers. This leads to the construction of a *value hierarchy* graph, a directed tree-like graph, the root of which comprises the final delivered service and the leaves represent services provided by known business entities. After the value hierarchy is complete, a *value exchange graph* has to be constructed. This graph is used to visually represent the e3value Value Network model, as described above. Deriving from the value exchange graph, *profitability sheets* are then generated to estimate the profit of the involved actors. Profitability sheets are calculated on a number of transactions accomplished within a certain time period. In this work, we rely on the profitability sheet concept to outline the results of our Cost - Revenue model discussed in section 3.3. According to Gordijn, as soon as the original model is complete, one can apply "model deconstruction operators" –i.e. a set of predefined transformation rules- in order to determine whether the Value Network can be transformed to optimize the actors' profits [26]. To enable e3value analysis e3value's development team provides a java application called e3editor [27]. The e3editor is capable of designing a value exchange graph –the visual representation of the Value Network- as well as generating profitability sheets in Excel file format based on calculation formulas filled by the user. The e3value framework uses an ontology to describe the Value Network; therefore, RDF files can be imported to the tool to automatically generate the value exchange graph. Note that in contrast with Verna Allee's approach that is primarily used to study existing Value Networks, the e3value methodology focuses on new, innovative e-services and facilitates the work of CEOs, marketers and even IT managers participating in the development process [25]. However, the detail level of the e3value model makes it difficult to model real-world, complex e-services. Additionally, this methodology relies on business process sequencing information that is not always possible for stakeholders participating in the early design stages to obtain making it more of an alternative to abstract business process languages. In the model we introduce in section 3.1, this kind of information is excluded from the Service Network level, so that it can be added optionally later on to enable the transformation of a Service Network to abstract business processes. By inference, e3value is a complete method for designing and analyzing e-services. It introduces highly usable concepts such as the profitability sheets while it manages to quantify value calculations by adopting a distinct value perspective.

2.2.3. The c3value and e3control frameworks

In addition to the e3value methodology, the c3value and e3control frameworks have been proposed [28][29]. C3value is based on the e3value model and focuses on strategic analysis. It studies Value Networks from the competitive advantage point of view and aims at exposing competitive values–called second-order values and complementary value objects- that allow a business model to stand out from the competition. As-is analysis, competition analysis and will-be analysis are performed to accomplish c3value's objectives. On the other hand, e3control is a modeling technique aiming to understand multi-actor service systems. In contrast to e3value, e3control examines the network assuming a "non-ideal behavior" of the involved actors and applies control mechanisms to avoid such opportunistic behaviors. The e3control methodology examines both the value and business process views of the network.

2.2.4. The Flow Graph and Value estimation (Caswell et al.) approach

A more formal model of Service Value Networks has been proposed by researchers at the IBM Watson Research Center (IBM WRC) and the Transformation Services Laboratory (TSL) in [2]. A flow graph is used, composed of a finite set of nodes $\{b_i\}$ that belong to *domain* B and a finite set of transfer objects $\{o_k\}$ that belong in the domain O. Domain B represents *economic entities* (business units) and domain O represents *offerings*. This model introduces *transfers* occurring between two nodes that contain a transfer object in the context of a *relationship*. Each node in the network consumes a set of offerings while producing other. To calculate the total value of the network they first define the set of business entities (end customers, producers, sellers etc.), and the set offerings and transfer relationships. Relations can have various properties, including the quantity of offerings, the revenues, the satisfaction index, trust and risks factors etc. The value of a node b_i in the Service Value Network in a certain period of time is calculated as the sum of the revenues derived from its relationships in the network at the time, minus the equivalent costs, plus the added value for *having* the relation. This extra value refers to the intangibles and the satisfaction index. The total value of the network is calculated to be the sum of the revenues of all sellers, minus the sum of product costs, plus the total value of the relationships. To support this framework, a tool named *VNT* has been developed [30]. VNT provides a range of Service Value Networks drawing functionality and enables the user to visualize and interact with Service Value Networks. The algorithm the tool uses to perform value calculations is tied to a specific Service Network example from the car-repair industry domain, explained in [2]. Therefore, the tool can only be used within the context of this example.

2.2.5. B. Blau's Service Value Networks

Benjamin Blau, has also proposed a model to describe Service Value Networks from the perspective of ICT development and Service-Oriented Architecture (SOA). Blau spots the lack of a Service Value Network definition from the information technology point of view and in [31] defines Service Value Networks as "…Smart Business Networks, which provide business value through the agile and market-based composition of complex services from a steady, but open pool of complementary as well as substitutive standardized service modules by the use of ubiquitously accessible information technology". The key elements in Blau's model are Service Providers, Service Offers, Service Requesters, Ownership Relations, Candidate Pools, Composition Relations and Complex Services. *Service Providers* supply *Service Offers* and to correlate providers with offers *Ownership Relations* are used. Alternative Service Providers offering similar services can exist in the network and therefore their offerings are grouped to *Candidate Pools* indicating that a Service Provider can be substituted on-demand. Incompatible service offerings can have *Composition Relations* with each other to indicate how a *Complex Service* is being composed. *Service Requesters* can create their Complex Services by requesting services from various Candidate Pools. A mathematical representation of this Service Value Network definition as well as an auction mechanism for service composition taking into account Quality of Service attributes is presented in [32]. In [33] C. van Dinther and Blau attempt to simulate this mechanism in order to analyze strategic behavior of Service Providers. Blau's approach is indeed the very first to examine Service Value Networks exclusively from the SOA point of view and as such, his methodology cannot be applied to networks involving human-related tasks. One could say that the method facilitates automated or semi-automated (web) service composition within a high level of abstraction. At the same time, since concepts such as the Candidate Pools and service on-demand are taken into account, Blau's Service Value Networks lie in between Service Ecosystems and traditional Value Network models; recall that originally Value Networks apply to business models where long-term relationships and trust issues occur.

2.2.6. Overview

Table 1 provides an overview of the methodologies discussed in this section. The "Key concepts" column enlists the key entities identified to form Service Networks as well as the key concepts of each approach. The "R-based" and "P-based" columns indicate whether the model focuses on Business Entities or on the roles they play in the Network ("Role-based" and "Participant-based" respectively). The "BP info" column shows if

sequencing or service composition information is incorporated to the model. To indicate whether a model applies to both web-service oriented and human-oriented Service Networks, we use the "Human-related" column. Finally, the goals that each methodology tries to achieve are shown under the "Goals" column.

Methodology	Key concepts	R-based	P-based	BP info	Human-related	Goals
ValueNet Works	-Participants -Transactions -Deliverables (tangibles / intangibles)	Yes	Yes	No	Yes	-Optimize value in existing networks
E3value	-Actors -Market Segments -Value Objects -Value Exchanges -Value Offerings -Value paths	No	Yes	Yes	Yes	-Design new services -Optimize revenues
C3value	-E3value's concepts -Second order value - Complementary Value Objects	No	Yes	Yes	Yes	-Perform strategic analysis
E3control	-E3value's concepts	No	Yes	Yes	Yes	-Control participants' behavior
Flow Graph Value estimation	-Business units -Offerings -Revenues -Customer satisfaction -Network-ness	No	Yes	No	Yes	-Optimize value in existing networks
B. Blau	-Service providers -Service offers -Ownership relations -Candidate Pools -Composition relations -Complex services	No	Yes	Yes	No	-Deliver complex services -Determine service prices

Table 1. An overview of the discussed Service Network models

2.3. Key Performance Indicators

Key Performance Indicators (KPIs) in general, are information items collected systematically in order to measure performance in systems that typically deliver services [34]. KPIs have been adapted by businesses over the last few years as a part of their monitoring process. In [35], Simmons defines business measures -in general- as

quantitative values that can be used for purposes of comparison. Measures could be compared to themselves, compared with target values –what we later on define as Key Performance Objectives (KPOs) in section 3.1.10- or evaluated along with other measures. In this context, KPIs are key business measures "directly related to the firm's strategy and are critical for its successful execution of its strategy" [36].

According to D. Parmenter [37], KPIs have seven characteristics: (1) KPIs are nonfinancial; financial indices comprise *result* indicators –e.g. daily profit- whereas KPIs disclose the causes of result indicators. (2) KPIs are measured frequently; an index measured on an annual basis cannot be of a key importance and it exposes performance of the past. KPIs are measures of the present –and even future- that are usually compared to past indices. (3) KPIs are acted on by the CEO and the management team; CEOs should be responsible for monitoring KPIs and pushing managers to deliver better results on a daily basis. (4) KPIs clearly indicate what action is required by staff –e.g. a high late flights index in an airline company shows that cleaners, ground crew, flight liaison officers etc. should do their best to speed up the flight preparation process-. (5) KPIs are measures that tie responsibility down to a team; the CEO should be able to contact someone and ask about a KPI's performance. (6) KPIs have a significant impact; they should affect one or more *Critical Success Factors (CSF)* –e.g. quality, customer satisfaction, sustainability etc.- and more than one Balanced Scorecards (BSC) perspectives –i.e. reports used to keep track of the execution of activities by staff-. (7) KPIs encourage appropriate action; a performance indicator needs to be tested to ensure it creates a positive impact on performance, in contrast with poorly thought-through measures that can lead to unwanted behavior.

A vast amount of KPIs has been proposed to measure performance in various industry domains. In [38], ways of applying KPI-based measurements in the Oil and Gas Engineering Project and Construction (EPC) industry are presented, while in [39] the authors number KPIs that should be used to integrate Environmental, Social and Governance (ESG) data into corporate performance reporting. A public open KPI database, where users are able to review, modify and suggest KPIs being categorized by process and industry can be found in [40]. Last but not least, APQC[2] -a member-based nonprofit organization that provides benchmarking and best practices- has introduced several KPIs within the context of the Process Classification Framework (PCF) [41]. PCF is a framework that organizes operating and management processes into enterprise-level categories and defines associated activities.

[2] http://www.apqc.org/

To better understand KPIs an indicative set of KPIs as found in [40] is presented in table 2. Please note that some KPIs apply to more than one sub-domains –i.e. '% of air carrier delays'-. Although KPIs should be measured over certain periods of time they usually do not explicitly define these periods; this is up to the evaluator –i.e. 'average occupation time of hospital bed' could be 'daily average occupation time of hospital bed'-. KPIs that do not comply with Parmenter's seven KPI characteristics are also listed, to demonstrate the deviation between several approaches. After all, it is up to the organization which KPIs it will adopt to better measure its performance.

KPI Name	KPI Description	Domains
Equipment availability %	Equipment Availability = Operating Time / Planned Production Time	Manufacturing, R&D & Maintenance > Manufacturing
Number of defects found over period of time	During the testing cycle, or even during the post release maintenance and support, the number of defects found/logged over a period of time is a good indicator of how product quality was changing over the Project Cycle or Product lifecycle.	Services, Technology & Consulting > Software engineering > Software quality assurance
Average occupation time of hospital bed	Average time (e.g. in hours) hospital beds are occupied by patients	Healthcare > Hospital
% of air carrier delays	Percentage delay of flights due to factors like cleaning, fueling, baggage loading and unloading, etc.	Aviation, Rail, Ship & Truck > Aviation > Airline, Aviation, Rail, Ship & Truck > Aviation > Airport

Table 2. A set of KPIs in several domains. Source: KPI Library [40]

3. Modeling Service Networks

In this chapter we propose a model for studying and analyzing Service Networks. We discuss the entities forming the Network and explain the way they interact. Our work focuses on bridging the gap between economists, business theorists and IT developers by providing a common language for understanding services and service delivery. At the same time we concentrate on the very services and define a core model that can be later on extended in many ways to facilitate Value calculations as well as transformations to Business Process Management models. In that sense, we diverge from previously presented models which either have a qualitative nature and appeal to CEOs and Business Theorists or are too complex and converge to Business Process models, targeting mostly to Business Process analysts and IT Developers.

The concept driving the Service Network theory presumes that any business can be modeled as a service [42]. Even when we deal with products, it is the *delivery of the product* that comprises the service offered to the End Customer. That is, in the car industry domain, the process of manufacturing a car can be modeled as a Service Network that incorporates all Business Entities working together to deliver a car – from parts suppliers and storehouses to the actual car manufacturer. Therefore Service Networks practically apply to all businesses, including raw producers and product manufacturers.

If we think of the car industry example a little more we can note an apparent peculiarity of the Service Network. Although many entities could participate in the Network and cooperate while offering and consuming several services, there exists a final single service that the Network provides: the car. This does not only occur in the context of the car industry example but also seems to apply to Service Networks in general. Although there is no limitation on how an analyst should model a Service Network - he is free to make any assumptions on the business model- and despite that no rule precludes the existence of many final -independent or not- services, the existence of a Service Network alone *implies* that there is a single service or a *bundle of services* that a key Business Entity delivers to an End Customer.

Although in real-world situations a Business Entity may offer various distinct and even unlike services while at the same time cooperating with lots of independent Business

Entities, within a Service Network a subset of these services is studied; the ones related to the "final" service that the Network delivers. The exact same thing applies to alternative service providers. Therefore, in our example, the car manufacturer can be in partnership with several car-shops owned by different business entities; in Service Network analysis though –as in classic Value Network analysis- we should rather model only one of these alternative service providers. To conclude, in an attempt to associate the already discussed concept of Service Ecosystems with that of Service Networks, one can say that a Service Network comprises an instantiated projection of a Service Ecosystem; with the difference being that the Ecosystem is more dynamic. Within a Service Ecosystem services are deployed, discovered, offered and consumed unfailingly, whereas in a Service Network lasting business relationships exist and evolution comes in a long-term period.

3.1. The Service Network meta-model

A first attempt to describe Service Networks towards the direction we follow can be found in [10]**.** Since then, several additions and improvements have been made in order to facilitate the constantly evolving theory of Service Networks. Concepts such as the *Key Performance Objectives –KPO*s- (we focus on Key Performance Objectives in section 3.1.10), the *Service* key concept as well as some special classes of Business Entities are introduced in this work. Before we discuss each Service Network entity in detail we should have a look at the Service Network meta-model, shown as a UML diagram in figure2.

A Service Network consists of *Business Entities* and *Services*. Business Entities are connected by Service Offerings and Service Consumptions. Service Offering relations correlate the offered Service with the source Business Entity (acting as service provider). Service Consumption relations describe the connection between the Service and the Service Consumer. Business Entities can be either *Participants* –the simplest form of a Business Entity-, *End Customers* –Business Entities that only consume services-, *Enablers* –special class of Business Entities that can offer services that *enable* the delivery of other Services (we discuss Enablers in section 3.1.4)- or Service Sub-Networks –Business Entities and Service Networks at the same time-. All Business Entities are represented by instances of their corresponding classes, and Service Offerings or Consumptions are instances of the *Offering* and *Consumption* classes respectively. Instances of Service Networks, Business Entities and Services have a name and are uniquely identified by an identifier. Business Entities play a Role in the network -e.g. "Ford" is the Business Entity with the Role being "car manufacturer"-. Services

could be goods or services, or a combination of both and they generate revenues. Revenues (modeled by the field revenue) are usually sums of money being gain for the source Business Entity (Service provider) and cost for the target Business Entity (Service consumer). All Business Entities have total costs and total revenues deriving from the services they offer or consume.

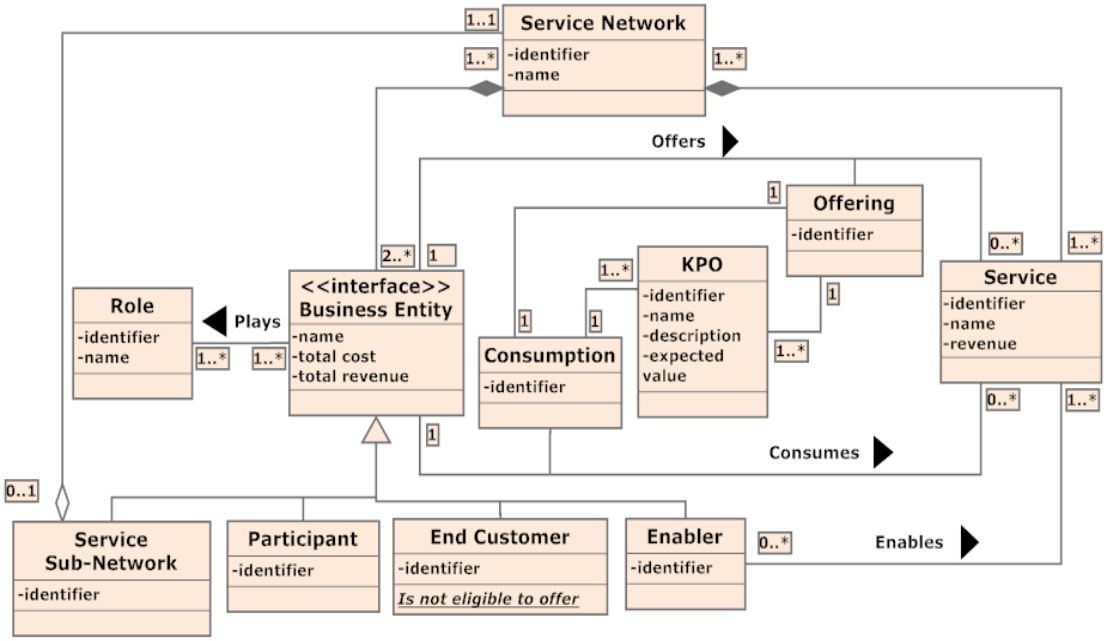

Figure 2. A UML diagram describing the Service Network meta-model

Generally, a Service Network model describes interactions among a set of Business Entities that take place over multiple, unrelated Business Processes. For each interaction both the source and target Business Entities may have business goals that they try to achieve. Therefore, Service Offerings and Consumptions are connected to Key Performance Objectives that describe these requirements. Key Performance Objectives are described by an identifier, a name, a description and the targeted metric value.

3.1.1. Business Entity

Business Entities are independent economic entities. They comprise the nodes of the Service Network and need to offer or consume at least one service in order to participate in the Network. In classic Value Network Analysis, Business Entities represent providers and consumers of functionalities that generate value in the Network as shown in [2][11]. In this work –as in the e3value approach discussed in section 2.2.2 – we assume that Business Entities only have costs and revenues that derive from the services they offer and consume within the Service Network. Consuming a service produces costs –an entity has to pay for the service- whereas offering a Service produces

revenues. As a result, our meta-model's Business Entities interface contains three key fields that identify a Business Entity: name, total cost and total revenue.

As one studies several examples of Service Network and Service Value Network analysis in the bibliography –e.g. [2][25][43][44][45] etc.- it becomes obvious that special classes of Business Entities can be identified. These classes have unique functional characteristics and need to be defined in order to facilitate both the visualization and analysis of Service Networks. We have identified four classes of Business Entities: Participants, End Customers, Enablers and Service Sub-Networks, all of which implement the Business Entity interface as expected.

Just before we expand on each Business Entity class we should note that some theories quote a fifth class of Business Entities: Raw Producers. Raw Producers are perceived to be starting nodes of the Network, Business Entities that do not consume but only offer services. Such entities do not have a distinct functional role in the Network that would differentiate them from Participants or Service Sub-Networks and would justify their inclusion in the meta-model. Moreover, in real-world scenarios Raw Producers are individual workers or resources that do not get modeled within a Service Network. They require a high level of detail that does not appeal to Service Network analysis; it would be infeasible for the analyst to model and study such a gigantic Service Network.

3.1.2. Participant

This is the simplest form of a Business Entity. We could say that any Business Entity not being End Customer, Enabler or Service Sub-Network should be modeled as Participant. In the bibliography, participants are also referred to as *Actors* [25]. Participants inherit their name, total cost and total revenue fields from the Business Entity interface and are identified by a unique identifier.

3.1.3. End Customer

End Customers are Business Entities that are not eligible to offer a service. As we discussed in previous section, Service Networks deliver at least one final service that is presumed to be the outcome of the Network. The Business Entity that consumes this Service is not a Business partner in the Service Network. Therefore we need a way of identifying the Business Entities that do not contribute to the Service composition. We call these entities End Customers.

The existence of an End Customer in a Service Network is optional. Although one would expect at least one End Customer to be present in any Service Network, this is not the

case. Think of the EBay business model. Their customers (sellers and buyers) actually offer a service to EBay by providing feedback on their experiences by dealing with other users. That way clients actually cooperate with EBay to improve the core auction service. And this is not the only known example of such a business model. Client feedback in general is essential to almost any business model as providers look forward to improve their products and increase customer satisfaction.

End Customers are uniquely identified by an identifier and as with Participants they inherit all Business Entities' attributes. As a result of not offering Services, the total revenue of an End Customer is always zero. In practice, except for facilitating Service Network analysis, End Customers prove to be very helpful in Service Network visualization as one can quickly identify the Service Network's delivered Service and / or the Business model that the Network adopts.

3.1.4. Enabler

A common problem that we came across from the very beginning of our study of Service Networks, is what we call the "triangulation problem". To demonstrate this we use a new but yet very simple example: the Amazon online book store. Amazon's original mission was to sell books worldwide. Customers could –and still can- buy books through a web interface, and the books got delivered to them by a shipping service provider – courier / postal service etc-. In this scenario we can identify three Business Entities: Amazon, Client and ,say, FedEx. We also distinguish two core Services: the book purchase and the book delivery. But how should we model this example? Clearly Amazon provides a service to the Client, which is the book purchase. Let's focus now on book delivery. To a certain extent, FedEx provides a service to the Client by delivering the book to her front door. On the other hand, Amazon needs a way of delivering books to her Clients in order for the book purchase to be meaningful. Normally, this should be modeled as a service provided to Amazon by FedEx. After all, Amazon is in partnership with FedEx, and most likely has earned a special price for being able to sell hundreds of books on a daily basis. Even if we overcome this ambiguity, we can still note a fundamental functional difference between FedEx and any other participant in the Service Network. FedEx is not a strategic partner of the Network as the composition of the book purchase service does not depend on the book delivery service. The role of FedEx in the Network is still crucial and could dramatically affect customer satisfaction, but FedEx most likely *enables* the Service delivery rather than affects its composition process. To identify entities that enable service delivery within a Service Network we

use the term Enabler. The services that enable service delivery are called Enablement Services.

Please note that the nature of a service determines if it should be modeled as an Enablement Service or not. In our example, let's assume that Amazon owns several storehouses. If we choose to include these in our model, FedEx will once again have to transport the books to Amazon before a customer's order gets processed. Now FedEx not only enables the delivery of books to the end customer but also enables the delivery of books to Amazon. This bundle of services could be modeled as a simple service that FedEx offers to Amazon, since Amazon have agreed with FedEx on a certain price for both services.

Enablers always interact with both service provider and service consumer. This also differentiates them from other classes of Business Entities that do not necessarily interact with any other Business Entity than the one they do business with. Another common class of Enablers could be intermediate payment service providers such as PayPal or Google Checkout.

The Enabler class implements the Business Entity interface –and therefore inherits all Business Entity's attributes- and has an identifier to uniquely identify it among other Enablers.

3.1.5. Service Sub-Network

Service Sub-Networks comprise Business Entities that have an internal structure of their own and contain an entire Service Network that provides and / or consumes services. Also known as "nested Service Networks", Service Sub-Networks have been an open problem in Service Value Network research field. Questions such as how does value flow into the nested Service Network and how the nested network's own value affects the top-level Service Value Network are yet to be answered. These questions are beyond the scope of this work as we do not deal with value; our model concerns only costs and revenues. But still, Service Sub-Networks are most likely to exist in a Service Network model and we should look into how costs and revenues flow inside and outside a Service Sub-Network.

In our Amazon example, for a book order to be processed several entities are involved. The accounts department offers accounting services, the customer support department offers support services and the computerization department offers relevant services as well. As we noted before, it is always up to the analyst how detailed the model will be and what services should be included or left out of the Service Network. In most cases

though, all these departments would not be modeled as a part of the core Service Network. If we need to study the internal structure of Amazon and try to find ways of reducing Amazon total costs by transforming the internal network, we would model Amazon as a Service Sub-Network and then model all these departments as an internal Service Network. The only restriction applying to Service Sub-Networks is that each service offered or consumed by a Service Sub-Network must also be offered or consumed by a single Business Entity inside the Sub-Network.

The Service Sub-Network class -as part of our meta-model- extends the Service Network interface and implements the Business Entity interface. Service Sub-Networks will also be discussed in section 3.2 where we will also show how a Service Sub-Network relates to the top-level Service Network.

3.1.6. Role

To discuss Roles we will use the EBay example. EBay offers auction services. Sellers can use EBay's well-known platform to sell items effortlessly while at the same time buyers can easily find what they are looking for. Both sellers and buyers need only to maintain an account in EBay's website in order to use EBay's services. There is also no restriction on who sells and who buys: anyone can be seller, buyer or both. Consequently, any Business Entity in EBay's Service Network could offer or buy items. In a specific Service Network instance Business Entity A could buy what Business Entity B sells, but in another Service Network instance it could be the other way around. In this occasion, we say that a Business Entity *plays* the Role of a seller and another one *plays the Role* of buyer; these Business Entities should be modeled as the Role they play in the EBay Service Network model.

Roles have a unique identifier, a name and connect to the Business Entity interface via the relationship 'Plays'.

3.1.7. Service

Services are the quintessence of Service Networks and can refer to both goods and services. This is what Business Entities exchange in the context of Service Networks. Services generate revenues. These revenues are presumed to be costs for the consumer and revenues for the provider. In the Amazon example, a customer has to pay 25€ for the last Harry Potter book. These 25€ are a gain for Amazon and a loss for the End Customer, although there is of course a gain in utility for the customer. Of course, due to the nature of the term Service, there might be situations where the revenues cannot be quantified. These services, also known as intangibles [22], might refer to information

and knowledge sharing, and cannot be contractual. In these cases, we assume that the Service produces zero revenues. Within the context of Service Networks these Services usually come in pairs, or co-exist in such a way that the network balance is not affected. To identify Services we use a unique identifier. Except from the revenue field, Services also have a name. Services connect to the Business Entity interface through the 'offers' and 'consumes' relationships.

3.1.8. Offering

"Offerings" is the class deriving from the "offers" relation between a Business Entity and a Service. Although by now the meaning of a Service Offering should be clear, please note that Offerings have a one-to-one relationship with Consumptions (the equivalent class for the consuming relation). This is to ensure that within a Service Network, for every Service a Business Entity offers, there exists another Business Entity that consumes it. This is another difference between the Service Ecosystem and Service Network viewpoints. In the Ecosystem there might be offerings not being consumed. This is not possible in Service Networks; Service Networks are used to model a certain business model and one needs to be connected to someone else to be part of that network.

Offerings indicate that target Service's revenues should be added to the Source Business Entity's total revenues. Since Offerings connect Business Entities with Services, they only have an identifier to uniquely identify them in the model.

3.1.9. Consumption

As with Offering, Consumptions is the class deriving from the "consumes" relation between a Business Entity and a Service. For every service consumption there must be exactly one offering in the network. Consumptions are being identified by an identifier attribute. For each Consumption, source Business Entity's costs increase by the revenues that the target Service generates.

3.1.10. Key Performance Objectives

Key Performance Indicators (KPIs) are business metrics used on the Business Process Management level for measuring the performance of business processes (KPIs were discussed in detail in section 2.3). There is usually some confusion when we talk about KPIs: we could refer both to the expected value and the measured value of the KPI of a business entity. In this work, we distinguish these two values by introducing the term Key Performance Objective (KPO). A KPO describes the targeted value, i.e. a target

performance value, and a KPI indicates the actual measured value. On the Service Network level, we use KPOs to describe the expected performance of underlying business processes from both the source and target Business Entities. For any given service, the service provider has her own business goals reflected to the KPOs that she will try to satisfy. At the same time, the consumer has some requirements that the service must meet and these should also be reflected to the KPO Model. As a result, in our meta-model Service Offerings are related to KPOs, and so do Service Consumptions.

KPOs are identified by a unique identifier and have a name and a description field. Since KPOs describe the targeted value of a KPI, an expected value field is also available (KPOs can refer to duration, money, efficiency etc and therefore the type of this field may vary).

3.2. Visualizing Service Networks

In figure 3, we provide a representation of a Service Network showing the relation created among two Business Entities. Business Entities are represented by circles and offering / consumption flows are represented through arcs. Service Offerings are depicted by solid arrows and Service Consumptions are depicted by dashed arrows. The latter, as expected, are used to show revenue flows. To keep Service Network notation consistent with the meta-model, offering and revenue flows always come in pairs; we could call them *dipoles*. Labels are used inside the circle and in the middle of the dipole to show the Business Entity's and Service's names respectively. To indicate the Business Entity class that an entity instantiates we use a small circle on the right of every Business Entity notation, containing the starting letter of each class: "P" stands for Participant, "C" for End Customer, "E" for Enabler and "SN" for Service Sub-Network. To indicate a service not being Enablement Service we use a yellow arrow in front of the service name.

Figure 3. The simplest form of a Service Network

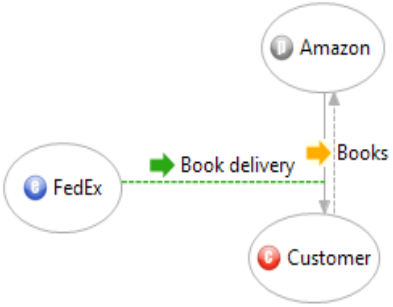

Figure 4. A simple Service Network with an Enablement Service

The Service Network shown in figure 3 is the simplest form of Service Networks that can exist: Amazon provides Books to the Customer. Amazon is an instance of Participant and

Customer is an instance of End Customer with Books being the Service. Please note that "Customer" refers to the Role that this Business Entity plays in the Service Network rather than the entity itself. If we move a step forward and decide to include the book delivery service, provided by FedEx, in our model we should end up with the Service Network of figure 4.

This figure demonstrates the triangulation problem as discussed in section3.1.4. FedEx provides an Enablement Service, enabling Customer to obtain the book he purchased. To discriminate between plain and enablement services, this time we use a green dashed arrow. At the same time, a thick green arrow is followed by the Enablement Service's name. In order to visually identify the service being enabled, we connect the enablement service's arc directly to the enabled service (i.e. Book Delivery service enables Book service). Please note that this way we still know the enablement service's revenue flows, as these are always costs for the enabled servicer's provider and revenues for the Enabler.

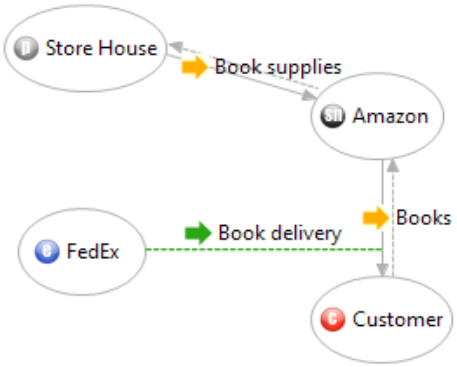

Figure 5. The Amazon Service Network example

In the examples studied so far we intentionally did not refer to Service Sub-Networks since special notation is required to show the revenue flows inside the sub-network. To introduce this, let's suppose that Amazon cooperates with a storehouse to provide it with Books that it will later on sell to the End Customer. We start with the Service Network of figure 4 and add the Storehouse Business Entity. Storehouse instantiates the Participant class, as it is not an End Customer (Storehouse is a Service Provider), it does not offer Enablement Services and its internal structure is not a concern in our network, so we choose not to model it as a Service Sub-Network. We name the service that the Storehouse provides "Book Supplies". But this time we want to look inside Amazon, and study its internal structure, so we will model Amazon as a Service Sub-Network. We end up with the Service Network shown in figure 5. Amazon is a cooperation with many departments dealing with customer orders, accounting etc. In fact, it is Amazon's own store department being responsible for processing and packaging a customer's order. To

keep things simple, let's assume that the accounting department inspects the packages, prints the invoice and finally deals with FedEx to ship the order. To represent this business model in our network we need to add two Participants inside Amazon's Service Sub-Network: The Storehouse Department and the Accounting Department. The first offers Packaging services to the latter. By now Amazon's departments are connected with each other but they do not interact with the top-level Service Network. As we discussed in section 3.1.5, every service offered or consumed by a Service Sub-Network must also be consumed or offered by a single entity inside the sub-network. Therefore, we need a way to show that Store House Department consumes the Book Supplies service and the Accounting Department offers the Books service to the End Customer. At this point we introduce the terms *Service Sub-Network (SSN) Input Port* and *SSN Output Port*. Each port represents a single service offered or consumed by the Service Sub-Network deriving from the top-level Service Network. In our example we should add an SSN Input Port for the incoming "Book Supplies" service and a SSN Output Port for the "Books" outgoing service. SSN Input Ports are notated with a green rounded rectangle, labeled with the incoming service's name. SSN Output Ports are depicted by a red rounded rectangle, labeled with the outgoing service's name. Ports are linked to Business Entities with single solid arrows, following the direction of the service flow. We simply call these *SSN Inputs* and *SSN Outputs* respectively. The resulting Service Sub-Network is demonstrated in figure 6. By using SSN Input and Output Ports, the Service Sub-Network manages to be consistent with the top-level Service Network model. Reversely, for a Service Sub-Network to be valid it needs to contain *at least one* SSN Input or Output Port. In this connection it may be remarked that the Service Network model is designed in a such a way that Service Sub-Network recursion is possible –i.e. a Service Sub-Network might incorporate another Service Sub-Network etc-.

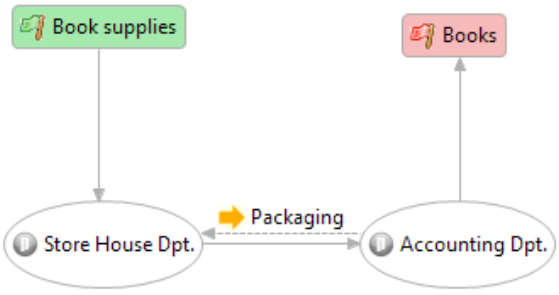

Figure 6. The Amazon Service Sub-Network

3.3. Simple Cost – Revenue model

The analysis of Service Networks is closely related to the *value* of the network. In section 2.2 we discussed several approaches for studying and analyzing Service (Value)

Networks, each of which employs diverse viewpoints on the subject. Benjamin Blau's work applies to software-based service systems [31]. Verna Allee uses sets of predefined indices to discover bottlenecks in a Service Network, following a more of qualitative-oriented direction [22]. The e3value approach calls up value path flows, somewhat related to business process management, and therefore appeals the most to IT managers and developers [25]. Calculations performed by Caswell et al. [2] quantify value, require utility functions and follow a domain-specific approach.

In this work we adopt a simple model for exploring the vitality of a Service within a Service Network. In an effort to quantify our results while retaining a domain-independent methodology we choose not to perform value calculations. However, since value is an inextricable ingredient of Service Network analysis, in the next chapter we demonstrate how value is created within a Service Network and discuss how the network's value is reflected to the customer's satisfaction index.

Each service delivery generates revenues. Service revenues are costs for the service consumer and revenues for the service provider. As a result, the profit of any Business Entity is calculated as follows:

$$Profit = \Sigma(IncomingServiceRevenue) - \Sigma(OutgoingServiceCost),$$

where IncomingServiceRevenue and OutgoingServiceRevenue refer to the revenues deriving from the Business Entity's consumed and offered services respectively.

To summarize the profits for each Business Entity in the network we use profitability sheets, as shown in [25]. Table 3 shows the profitability sheets for Amazon, based on the example of figure 5. Keep in mind that the prices are illustrative and refer to -as realistic as possible- averages.

Name:	Amazon		**Entity Type:**	Participant	
Total cost:	18€		**Total revenue:**	23€	
Business Entity	**Service consumed**	**Service cost**	**Service offered**	**Service revenue**	**Enables other service**
Store House	Book supplies	15€	-	-	No
FedEx	Book delivery	3€	-	-	Yes
End Customer	-	-	Books	23€	No

Table 3. Profitability sheet for the "Amazon" Business Entity

4. Enhancing Service Selection using Service Networks Analysis & Requirements-based service discovery

Modeling a business landscape as Service Network, allows, on the one hand, calculating the value gained by a single partner when joining the collaboration network. On the other hand, a Service Network perspective gives the possibility to measure the value of the whole network and potentially increase the satisfaction of the end consumer. When it comes to software, its potential value will be, to a large extent, determined by its incorporation into service offerings and the ability for these services to participate in existing and new Service Networks. To enable Service Networks however, more suitable combination of free-text search techniques with ontology-based search techniques are necessary. This combination will furnish Service Networks with structured discovery as well as searches with a retrieval style suitable for discovering unstructured information.

In wider spanning service ecosystems, service providers may offer functionally replaceable services that differ in their quality characteristics. Service ecosystems should therefore explicitly support the negotiation process, reducing non-critical human involvement and providing decision-makers with the information they require to formulate and assess service offers. In particular, automated support for negotiation over services is needed for comparing requirements of prospective service consumers against capabilities and qualities of available services. This calls for techniques and tools supporting the detection of potential service guarantee violations that will ultimately result in an improved service selection process.

In this work, we have developed a process to (1) discover candidate services based on quality of service characteristics and use them to dynamically identify various classes of service providers to create new Service Networks within a service ecosystem, (2) exploit newly created Service Networks to improve the service discovery and selection process.

4.1. Value in Service Networks

Within a Service Ecosystem, value is co-created every time there is service delivery from a service provider to a service consumer. The service provider derives value from any compensation or revenue he gets and the service consumer's satisfaction (if any; satisfaction is usually a good indicator of the willingness of the consumer to ask for

more service in the future). The service consumer derives value from the usefulness of the service delivered and possibly from its use further down the line, as the service consumer herself composes her own offering.

The consumer's utility function (which measures the usefulness – utility – of a service) is closely connected to the customer satisfaction. Various methodologies have been proposed that measure the customer satisfaction; we adopt the methodology of the American Customer Satisfaction Index (ACSI) [46]. ACSI is a customer based measurement system for evaluating and enhancing the performance of firms, industries economic sectors and national economies. It uses customer interviews as input to a multi-weight econometric model [47]. By applying ACSI to the service discovery process we are able to monitor and select services more effectively using consumer-related information such as customer demands and feedback. An increase or decrease of customer satisfaction will encourage service providers to react to user feedback, thus enabling a more competitive market.

As Service Networks assume that services can be matched and retrieved, the next section describes a requirements-based service discovery environment that discovers and selects services using service qualities. Recall that Key Performance Objectives are already used on the Service Network level to describe service requirements for both the service provider and consumer

4.2. Discovering and Selecting Services based on Quality of Service

In order to form and execute service queries from requirements specifications, new tools and techniques have been developed -described in [48]-. Service consumers form service queries from requirements specifications to retrieve web services compliant with the requirements. Descriptions of retrieved services are presented to service consumers who use them to refine and complete requirements to enable more accurate service retrieval and ultimately select the services that best satisfy their requirements. The service discovery environment has 4 main components: (1) Service Registries – a federated and heterogeneous mechanism for storing both the service implementation that applications invoke and one or more facets that specify different aspects of each service including service qualities, cost/commerce and description [49]; (2) the UCaRE Requirements Component, which supports web-enabled specification of requirements and use cases, and formulation of service queries from these specifications; (3) the Service Querying Component, which uses service queries to discover web services with different types of similarity, and (4) the Service Explorer Component that displays the descriptions and specifications of retrieved services to enable service consumers to

understand, select between and use these services. A full description of the 4 components is provided in [48].

To support the process of requirements-based service discovery a means to represent service semantics has been developed. Relevance feedback may be used to inform users' formulation of requirements based on representations of service semantics held in service registries. Crucially, this may include information about non-functional service characteristics, namely Quality of Service (QoS). Quality of Service has been identified [50] as a key element in the wider uptake of web services, particularly in a competitive market in third-party services [51]. Here, the consumer has no control over the service provision and implementation, and so assessing Quality of Service is vital in establishing trust in a service. It is therefore desirable for the service consumer to be able to clearly state Quality of Service requirements and identify the level of service compliance with those requirements. With the requirements-based service discovery tools and techniques it is possible to get an early indication of the quality of available services by matching non-functional requirements (NFR) to service qualities during service selection [52]. To overcome the quality translation problem – i.e. being able to compare requirements qualities and service qualities that are stated using different metrics and units – the service discovery environment makes use of the Quality of Service Ontology QoSOnt [53], to provide a common understanding between service specifiers and consumers and allow reasoning about queries as a means to tolerate inexact matches between QoS queries and specification in terms of metrics and units. This simplifies the problem of reaching a common understanding of quality and allows translation where providers choose to specify quality differently [52]. By using our existing service discovery environment we are able to enhance Service Network analysis by matching Key Performance Indicators to service qualities. In the remainder of this chapter we describe the approach and demonstrate it with an example.

4.3. KPI-driven Service Discovery Process

The process is iterative and incremental as shown in figure 7. The process consists of five main steps:

1. A service ecosystem comprised of SNs is analyzed by identifying the entities participating in the ecosystem, the roles they play as well as the offerings and revenues that emerge from the analysis [2].

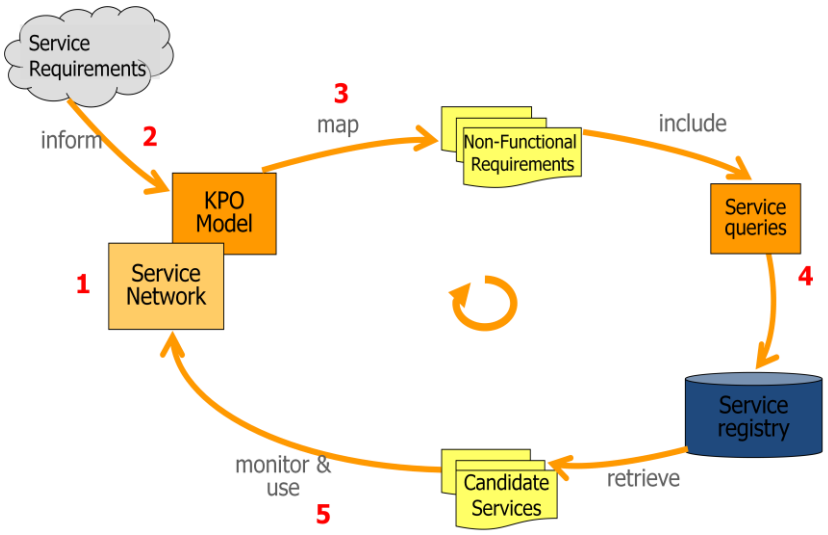

Figure 7. Iterative and incremental KPI-driven service discovery process

2. The KPO Model is generated. In order to do so, it is necessary to acquire the following resources:

   i. *Knowledge of Service Provider's strategy* [10]. Each organization has its own business goals reflected to the Key Performance Objectives that it will try to satisfy.

   ii. *Service requirements*. This applies to the service consumer. He has some requirements that the service must meet and these should also be reflected to the KPO Model. These requirements are not to be confused with the NFRs that we use later on in the process.

3. In order to identify the level of service compliance with the KPO Model using the existing requirements-based service discovery environment, we need to map each KPO to its corresponding NFR. The provision of a semi-automated way to manage the mapping process is of key importance to assist the analyst in converting the KPO Model into a set of NFRs. It is through commitment to the same ontology (QoSOnt) in the service specification and discovery tool that comparison between service qualities is made possible. Therefore, it is appropriate to use the same ontology for the conversion of the KPO Model. This means that if the underlying ontology is altered then these changes will automatically be reflected in the conversion mechanism allowing a correct comparison between requirements and QoS characteristics, and in extension between KPOs and QoS characteristics. In practice this does mean that some level of agreement on a QoS vocabulary is a prerequisite. However the use of

ontology easily allows the expression of new concepts in terms of other existing concepts. To achieve such a conversion, we plan to develop a wizard-like tool that guides the analyst throughout the process based on the QoSOnt Ontology. Firstly, a KPO is automatically mapped to a requirements type based on predefined mapping rules. Secondly, the analyst selects a QoS metric from a list of QoS metrics directly taken from the QoS Ontology that relate to the previously mapped requirement type. The final development of the mapping mechanism is part of the future work

4. Once mapped, each generated NFR is included into the requirements specification. One or more service queries are then generated from the specification, passed on to EDDiE, the service discovery engine, and used to discover candidate services based on QoS.

5. As specific services are selected – among the set of candidate services – their performance is measured by calculating the KPIs using tight and/or loose service monitors. Here, we assume the existence of a third party monitor that analyses the conformance of a web service to KPIs. Please note that we also assume the correctness of the reported conformance results. At the same time service registries are monitored using the requirements-based service discovery environment for newly available services that conform to the KPO Model. The KPI monitoring in-formation improves service discovery and selection by (1) filtering out services (whose specified QoS does not match the measured KPIs) that do not comply with the KPO Model, and (2) replacing such services with newly published candidate services that do comply with the KPO Model. As time progresses, some links (dipoles) occur more frequently than others, indicating a preference of the consumers for some of the services offered. In order to increase the overall customer satisfaction, we monitor how the changes influence the service customer's opinion on each consumed service by calculating the overall customer satisfaction using the American Customer Satisfaction Index (ACSI). ACSI regards satisfaction as a latent variable, i.e. satisfaction is measured indirectly. It consists of 3 measurable variables that are used to capture the degree of customer satisfaction for the consumption of a service [46]. The overall satisfaction index is represented as a score ranging from 1 to 100. It is worth noting that the complete ACSI model as described in [46] takes into consideration additional aspects such as the expectation, the loyalty and the complaints of customers.

4.4. Real-World Example: In-Car Service-centric System

In this section we demonstrate key elements of the process using an example taken from the automotive industry [54]. The example explores how services in automotive systems can enhance Service Network analysis.

A car original-equipment-manufacturer (OEM) is looking to use services in the automotive domain for various purposes. One is to improve customer satisfaction by providing the car with an advanced and customizable on-board device, capable of providing the driver with many specialized services. The second is to improve the customer relationship management process, and consequently customer satisfaction, by providing services to the driver via an on-board device. The on-board device is able to invoke remote services by connecting to a server managed by the OEM's service centre. The service base contract called *Smart Referral Service*, subscribed by car owners, includes the following services:

1. *Navigation*. Textual and vocal information about any destination needed by the driver.

2. *Weather information*. Weather forecast on the basis of the current car position.

3. *Traffic information*. Traffic events related to the area in which the car is localized.

4. *Remote maintenance*. Early diagnosis including ad-vices about detected part faults, and spare part avail-ability of the nearest authorized parts supplier and workshop.

In figure 8, we show the flows of offerings among the various partners (shown as circles) in the Smart Referral service ecosystem. Offerings and payment flows are represented by arcs. Please note how this notation is tightly connected to the Service Network notation discussed in section 3.2**.** The main actors are the *Car Owner* (service consumer), and the OEM who offers on-board services to its customer, i.e. the Smart Referral Service. This service is a bundle of services combining already available services offered by service providers. All remaining partners in the service ecosystem indicate the roles of each service provider whose service will be available through the system. These roles include the *Traffic Report Provider*, the *Weather Forecasting Provider*, the *Parts Supplier Provider* and the *Navigation Provider*. All these businesses pay a fee to the OEM who acts as a service broker. The customer satisfaction index is of key importance to the determination of the overall value of the service system; a

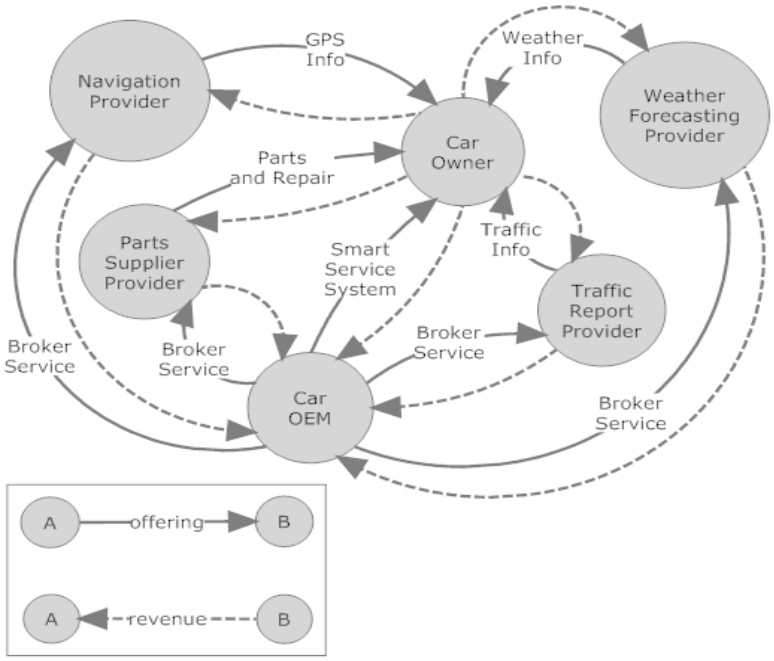

Figure 8. Service ecosystem of Smart Referral Service

declining index lowers service consumption, while an increasing index raises service consumption. For instance, delays in providing a weather forecast result in longer waiting times as perceived by the car owners, thereby lowering the car owners' satisfaction.

4.4.1. The route calculation Scenario

To apply our model to the automotive example we will focus on the *route calculation scenario* and therefore focus on the interaction between the *OEM*, the *Car Owner* and the *Navigation Provider*. The scenario is as follows: A driver is driving his car. The driver needs to determine the route to a specific address. The driver activates the car's on-board navigation service to provide the route and visualize the best route between the current car position (GPS coordinates) and the specified address. Based on the car owner's contract and profile, the service sends a suggested list of route information to the driver who then selects a route.

In the example we will demonstrate one simple iteration of the process depicted in figure 7. We apply the process described in section 4.3 to the single offering of *GPS Information* between the *Navigation Provider* and the *Car Owner* (see figure 9) and provide early performance insights of the process. Additionally, we will check the accuracy of the process and how it can lead to new emerging Service Networks and improve the service ecosystem as a whole.

Figure 9. Service Network between Car Owner and Navigation Provider

4.4.1.1. Generating KPO model

In order to generate the KPO model for the route calculation scenario we consider example KPOs for the car owner. Since our focus is on service consumer requirements KPOs that describe the objectives of the service provider are not considered in this context. Assuming the existence of a service requirement specification (elicited using e.g. questionnaires), the analyst uses these service requirements to generate the following example KPOs (expressed in natural language):

1. *Response Time* – How long does the service need to calculate and display the route?
2. *Costs* – What are the total costs for using the service for each request?
3. *Reliability* – What is the relation between the total number of successfully performed route calculations and the total number of requested route calculations?
4. *Availability* – How many times was the service available when requested with respect to the total number of service requests?

Table 4 lists the above example KPOs for the scenario, including their calculation functions and the expected target values used to compare the quality of available services. Note that as we try to demonstrate the application of our approach, the target values presented in table 4 are example values that have not been derived from concrete service requirements.

KPO	Metric Calculation	Target Value
Response Time	t(responseTime)	<10 seconds
Costs	c(request)	<10 €
Reliability	(successfulRoute / totalRoute) * 100	>99%
Availability	(respondedRequests / totalRequests) * 100	>90%

Table 4. Example KPOs for the route calculation scenario

4.4.1.2. Mapping KPO model to Non-Functional Requirements and QoS

Next, the KPOs are mapped to their corresponding requirement types and QoS metrics using the approach as described in section 4.3. For instance, the *Response Time* KPO is mapped to the *Performance* requirement type and in turn mapped to the *MeanTimeToComplete* QoS metric based on the list of QoS metrics directly taken from the QoS Ontology. Table 5 shows each mapped KPO from the navigation example.

KPO	Requirement Type	QoS metric
Response Time	Performance	MeanTimeToComplete
Costs	Cost	Cost
Reliability	Reliability	ProbabilityOfFailureOnDemand
Availability	Availability	AvailabilityAsPercentageUptime

Table 5. Mapping KPOs to requirements and QoS

4.4.1.3. Discovering Services using Requirements queries

A requirements-based service query is then formed consisting of the set of converted NFR and fired at service registries to discover services that are compliant with the specified requirements. The service discovery environment presents and explains each discovered service to the service consumer who either accepts or rejects the service as relevant to the KPO Model. To keep the example simple, we assume the discovery of four (4) candidate services – *Navigation1*, *Navigation2*, *Navigation3* and *Navigation4* – that match to a service query consisting only of the *Availability* KPO. Table 6 shows the corresponding *AvailabilityAsPercentageUptime* metric values for each discovered example service. Table 6 also shows the converted availability value that QoSOnt inferred for the specifications that do not directly specify their availability. This demonstrates the feasibility of the service discovery environment to filtering candidate services based upon quality independent on how a service provider specifies services' Quality of Service properties, and how a service consumer specifies QoS requirements [52].

Service Name	Specified metrics	Specified values	Percentage uptime
Navigation1	Percentage uptime	91.98 %	91.98 %
Navigation2	Mean Time To Recover (MTTR) Mean Time Between Failures (MTBF)	1 Hour 42 weeks	99 %
Navigation3	MTTR MTBF	60 Minutes 294 Days	95%
Navigation4	Percentage uptime	93 %	93 %

Table 6. Expected and actual outcomes of the discovery activity

4.4.1.4. Monitoring Services based on KPIs and Satisfaction Index

We then monitor the ecosystem's behavior for several periods of time using the candidate services and calculate the KPIs based on monitoring results. In addition, during the same period we repeatedly measure the satisfaction index as described in section 4.3. The resulting KPIs should be an accurate, honest reflection of the service efficacy in delivering the outcome based on its QoS. Table 7 shows examples of measured values for the Availability KPI compared with each service's QoS metric value. As shown in the table, there are three possible outcomes: (1) the KPI is above the specified QoS value and therefore satisfying the KPO, (2) the KPI is below the specified QoS value but still above the KPO and therefore satisfying it, and (3) the KPI is below the specified QoS value and consequently not satisfying the KPO.

Service Name	Specified QoS Value	Availability KPO	Availability KPI
Navigation1	91.98 %	>90 %	80 %
Navigation2	99 %	>90 %	95 %
Navigation3	95%	>90 %	95 %
Navigation4	93 %	>90 %	96 %

Table 7. Availability KPI monitoring results compared to the corresponding KPO and QoS metric of each candidate service

4.4.1.5. Impact on Service Ecosystem and Service Selection

The results show that certain service providers might not deliver the service as promised since the KPIs do not match with the specified QoS. Note that we assume but cannot guarantee the correctness of the monitoring process based on the resulting KPIs as this depends on the KPI monitoring technique used by the OEM. Collecting such information enables the OEM to calculate the rate of deviation between expected and measured quality values. Each iteration of the process computes new KPIs that indicate the level of compliance with the KPO Model. Results are incrementally used to improve the service discovery and selection process by filtering out services that do not satisfy the KPOs as specified by the service consumer. At the same time, newly published navigation services are discovered and monitored for KPO compliance. As a result, services from various navigation providers either enter or exit the service ecosystem depending not only on their specified QoS values, but also on their measured performance. Consequently, an improved service selection process increases the satisfaction index after each evaluation period as well as the quality of the overall service ecosystem.

Service consumers begin to build strong relationships with service providers resulting in new Service Networks that did not exist before. Patterns of interaction thus occur, as some links (dipoles) are almost always active between a service provider and a service consumer. For example, *car owner X* prefers to use the *Navigation4* service as the navigation provider and *Weather1* service as the weather forecasting provider, whereas *car owner Y* might use the *Navigation2* service and *Weather2* service instead.

5. Designing the Service Network Analysis & Prediction Tool

In the line of this thesis, we have developed a software application that enables modeling and analysis of Service Networks, as discussed in chapter 3. The tool is called *Service Network Analysis & Prediction Tool* (SNAPT).

Despite the fact that we have already expanded on our Service Network model and the Cost-Revenue analysis, a little is said about Service Network prediction. Prediction typically occurs in the context of simulations that expose the network's expected behavior in a given period of time. Several already established and specialized simulation software tools exist –e.g. VenSim [55], iThink [56] etc-. We plan to allow SNAPT to communicate with such tools in order to fulfill the prediction aspect of Service Networks. However, this is beyond the scope of this work and is further discussed along with other future challenges and additional functionalities of SNAPT in chapter 7.

5.1. Model-driven architecture (MDA)

Since Service Network theory is constantly evolving, it is possible that the Service Network model will be a subject to future revisions. Therefore, a major challenge we came across when designing SNAPT was to allow future modifications of the underlying model without affecting –or affecting the less- SNAPT's already implemented functionality. To achieve this, we have developed SNAPT following a Model-Driven Architecture rather than a traditional monolithic application approach.

Model-Driven Architecture (MDA) was introduced in 2001 by the Object Management Group (OMG)[3]. OMG has also contributed several well-established standards such as the Common Object Request Broker Architecture (CORBA) [57], Unified Modeling Language (UML) [58], XML Metadata Interchange (XMI) [59]-. MDA is a software architecture that enables the use of system models in the software development process [60]. According to OMG, MDA is defined as "…a way to organize and manage enterprise architectures supported by automated tools and services for both defining the models and facilitating

[3] http://www.omg.org/

transformations between different model types" [61]. MDA is dominated by four principles [60]:

1. Well-expressed models are of major importance to understanding systems for enterprise-level solutions.

2. A series of model transformations, appropriately organized into an architectural framework of several layers and transformation can lead to efficient systems development.

3. A formal infrastructure for describing models as a set of meta-models facilitates integration and transformation among models, and is the basis for automation through tools.

4. In order to accept and adapt this model-based approach, industry standards are required to provide openness to consumers, and infuse competition among vendors.

In alignment with these principles, OMG has defined three different viewpoints: The *computation independent*, *platform independent* and *platform specific* viewpoints, each of which has its corresponding model [62]. A *platform dependent* model is also defined that is rather technical and refers to the services a platform provides.

In the computation independent viewpoint, the focus is on the system's environment and the system requirements. Structural as well as processing information is usually undetermined or not represented. The corresponding model, then, is a computation-independent view of the system. The computation-independent model (CIM) is commonly used by business analysts, as long as no knowledge on the application development process is required to produce and understand it. Therefore, the CIM is an important bridge over the gap between the business analysts and the engineering developers as far as MDA is concerned. CIM describes the functions involved in the system at a high level of abstraction and covers the interactions between the processes and the responsibilities of each worker—human or machine. As a result, one could say that anyone having an understanding of business processes is also able to understand a CIM.

This platform independent viewpoint expands on the operation of the system, by exposing the pieces of the specification that should not change from one platform to another. The corresponding platform-independent model (PIM) does not go into detail about what is necessary for several platforms; it actually ignores the operating systems,

programming languages or hardware aspects of the system. As a result, PIM represents the business model that the information system should implement.

The platform specific viewpoint incorporates the platform-independent viewpoint along with specific platform details. The corresponding platform-specific model (PSM) shows how a system can use a specific type of platform. PSMs can be expressed using UML [58], Domain-Specific Languages (DSL) or general purpose languages –e.g. Java, C++, Python etc-.

According to OMG, when talking about a "System", we may refer to a program, a single computer system, a federation of systems, an enterprise or even an enterprise consortium. Additionally, a platform may refer to a generic platform type –e.g. object, batch or data flow-, a technology specific platform type –e.g. CORBA, Java 2 Enterprise Edition (J2EE) etc- or a vendor specific platform type –e.g. IBM WebSphere software platform [63], Microsoft .NET [64] etc-.

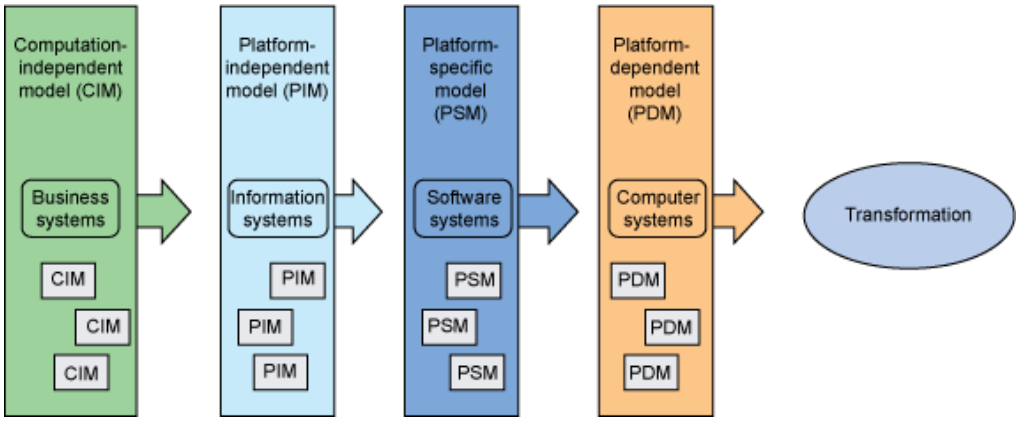

Figure 10. The basic MDA Process. Source: [ref] (62)

Figure 10 demonstrates how MDA models and viewpoints build on one another. Business system needs and requirements are modeled during the CIM process. The supporting processes of information systems are then modeled. The information-system modeling created during the platform-independent viewpoint and modeling phase is then supported by software systems modeling developed during the platform-specific phase; finally, computer systems modeling in the platform dependent model support the needs of the software systems. You can see how multiple inputs are received to create the overall model in a specific phase [62].

5.2. Eclipse Graphical Modeling Framework (GMF)

The Eclipse Graphical Modeling Framework (GMF) is an open-source Domain Specific Language (DSL) framework facilitating the development of graphical editors [65]. GMF

is based on the Eclipse Modeling Framework (EMF) [66] and the Graphical Editing Framework (GEF) [67]-both GEF and EMF frameworks are part of Eclipse's Modeling Project-. GMF helps in defining the domain model, its properties, as well as relationships among them (the domain model's own meta-model as well as the code deriving from the first are both part of the EMF's responsibilities within the framework). A set of editor-specific meta-models such as the graphical model, the tooling model and the mapping model are later on defined. These models are finally automatically transformed to provide graphical representation of the domain model and its constraints; this graphical representation is achieved using GEF.

GMF's two main components are the *GMF Tooling* and the *GMF Runtime* [68]. The set of models used to generate the graphical editors as well as appropriate tools to transform them are part of the GMF Tooling. On the other hand, the GMF runtime is responsible for bridging the EMF and GEF runtimes, offers reusable components for graphical editors and contributes a standardized model to describe diagram elements while separating them from any semantic model information. Figure 11 shows the dependencies between EMF, GEF and GMF runtimes as well as how GMF Tooling is used to generate graphical editors.

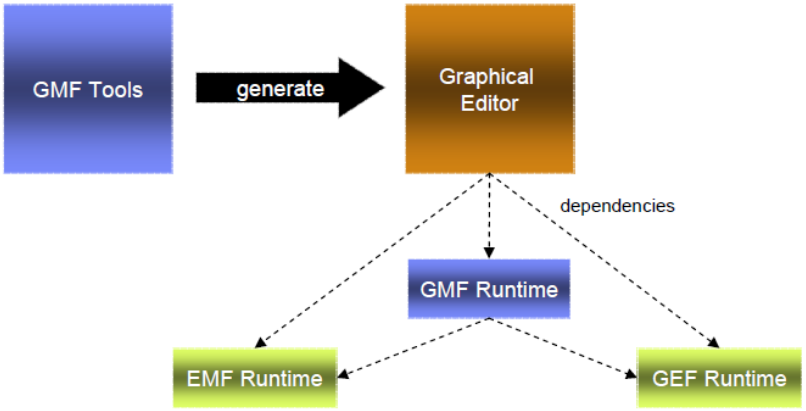

Figure 11. GMF dependencies between the various runtime & tooling components. Source: [ref] (68)

5.2.1. GMF Runtime

GMF runtime's architecture allows for reusable components used in graphical editors to facilitate common actions. It offers a set of geometrical shapes –e.g. circle, polygon rectangle, poly-line etc.- used to represent both nodes and edges in a graphical diagram. Graphical components such as action bars and connection handlers operate on the diagram editor's graphical user interface to let the user quickly create new diagram nodes and connections. Additionally, typical components like toolbar contributions, editing and clipboard actions are also implemented. Except for providing graphical

components, the GMF runtime supplies a standardized model to define diagram elements. This model holds information regarding the nodes –i.e. layout constrains, a node's size and location etc-, the edges –i.e. bend-points, source and target anchor points- and the styles –i.e. fonts, sorting information, line styling etc.- of each diagram. Another task that GMF runtime accomplishes is to bridge the EMF and GEF command infrastructures. Each action performed in an Eclipse-based application is added to a so-called "command stack", which is responsible for the command's execution and manages the undo / redo stacks. Since EMF and GEF have their own command stacks, the GMF runtime is responsible for unifying the underlying implementations and keep both graphical (GEF) and semantic (EMF) models synchronized. For example when a user deletes a node from the diagram, all graphical components, GMF information as well as the semantic element must be deleted. So both EMF and GEF's delete commands must be executed and the application should also be able to create undo and redo contexts for the delete action. Finally, the GMF runtime provides extensibility through extension points. Extension points are Eclipse's internal mechanism for interpolating code with additional functionality through XML declarations [69]. In GMF, all layouts, decorators, diagram model, diagram elements and palette can be extended. By inference, one could say that the GMF Runtime is a *set of frameworks* to help the development of Eclipse graphical editors [70].

5.2.2. GMF Tooling

GMF tooling is the component responsible for fulfilling the MDA aspect of GMF [71]. It consists of several models complementing EMF's domain and generation models to deliver a fully functional graphical editor. The models that constitute the GMF tooling component, their interdependencies as well as the generated code packages are depicted in figure 12.

The process starts with the domain model, a meta-model that uses conventional object oriented concepts like classes with attributes and references to other classes to describe the semantic model. Just like in any object-oriented approach a class can extend any other classes, inheriting their features. Special data enumeration types can also be defined to be later on used as attribute types. EMF's domain language is named *Ecore*. Ecore models can be generated using UML, XML schemas or even modeling tools and appropriate Eclipse editors.

To enable generation of basic code for expressing and manipulating these classes the *generation model (GenModel)* is used. In MDA terms, the GenModel is a platform-specific model (PSM) expressed in a Domain-specific language that facilitates the transformation

of the Ecore model –i.e. the Platform-independent model- to executable Java code. The GenModel holds information such as the names of the Java packages and classes to be generated; it also determines whether attributes should be editable or not, provides sorting information etc. Once the user is ready to generate the code, the following will derive from the GenModel (as described in [66]):

1. *Model* - Java interfaces and implementation classes for all elements in the Ecore domain model; it also contains a factory –i.e. a Java class responsible for instantiating the domain model classes- and package -.i.e. a class regarding meta-data objects- implementation class.

2. *Adapters* - implementation classes (called ItemProviders) that adapt the model classes for editing and display.

3. *Editor* - a properly structured editor conforming to the recommended style for Eclipse EMF model editors. This EMF editor is optionally generated as far as the GMF workflow is concerned.

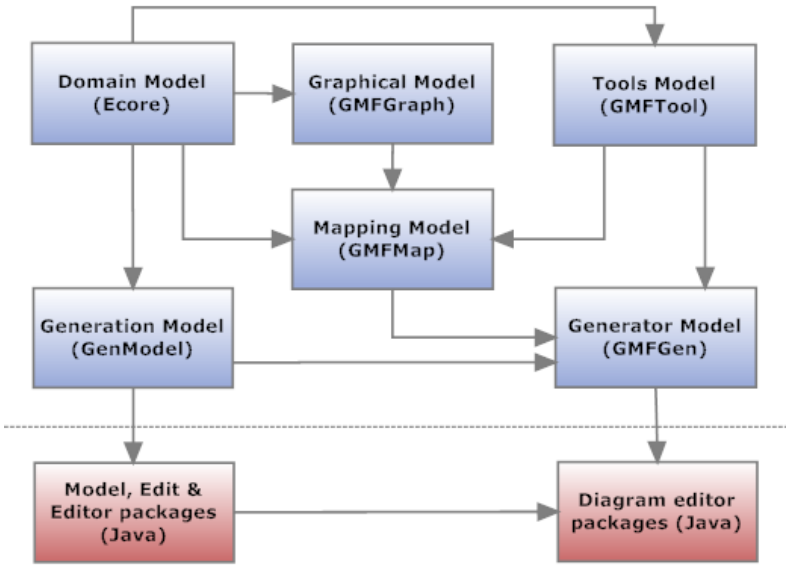

Figure 12. GMF Model interdependencies and code generation

The *graphical model (GMFGraph)* defines the symbols used for visually representing object classes (what will become the nodes of a diagram) in addition to labels for displaying and editing attribute values on the diagram. Moreover, the graphical elements of the diagram's connections along with sub-nodes (objects contained in compartments within other nodes) can also be defined. A wizard can be used to automatically generate a basic graphical model from the domain model; the chances are that this model will serve as a template for further manual customization (describe custom figures or add decomposition information).

The *tools model (GMFTool)* defines the tools for creating new nodes and connections in the diagram editor. GMFTool is typically used to define the editor's palette. Except from object creation tools, GMF can also generate standard common tools such as zoom-in / out and selection tools, in addition to allowing a user to define his own custom tools. To further customize the tools palette, one may also group the tools and define his own icons. A wizard can also be used to automatically generate the GMFTool model from an Ecore model.

The *mapping model (GMFMap)* connects the Ecore, GMFTool, and GMFGraph models, defining which node and connector symbols are used for representing which domain model classes and which tools are used for the creation of these objects. Furthermore, the GMFMap model can be used to define diagram-level constraints. If, for example, we need to forbid a node from connecting to itself, the GMFMap model is the right place to provide this information. As with previous models, a wizard can be used to automatically perform basic mappings, using as input all three Ecore, GMFTool and GMFGraph models.

Finally, the model responsible for generating the graphical editor's Java code is the *generator model (GMFGen)*. This model, as expected, holds code-specific information. Java class names and prefixes, plug-in and model IDs, menus, toolbars, application actions –e.g. close window, copy, paste, select etc- are all described in the GMFGen model. In most cases, the GMFGen model will be automatically generated from the GMFMap mode and will not need any further customization. However, this was not the case with SNAPT. SNAPT is a complex application that in fact uses two different graphical editors deriving from the very same domain model: one is used to edit Service Networks, and the other for Service Sub-Networks. As these editors need to smoothly interact with each other, several issues were exposed (we expand on these issues in chapter 6); dealing with these issues requires to interfere with the automatically generated GMFGen model. To enable automatic transformation of a GMFGen model to a new GMFGen model based on transformation rules we use the *GMF Tools Project*.

5.3. The GMF Tools project

The GMF Tools project offers a collection of development tools and extensions applying to both the GMF runtime library and GMF's code generator. From now on, we refer to the GMF Tools Project using the acronym *GTP*, as it must not be confused with GMF's tooling component or the GMFTool model described in the previous section. GTP is an open-source coding project aiming to render development with GMF more efficient and productive [72].

GTP interferes in GMF functionality on the model level. To accomplish this, it applies model-to-model (M2M) transformations to the automatically generated GMFGen model, to deliver a new, customized GMFGen model. The transformation mechanism serves a dual purpose: (1) It embeds template extensions to GMF so that generated code will be able to handle GTP's custom code. (2) Allows GMF users to add their own transformation rules into the M2M templates. While in the process of developing a GMF editor, a minor change to any of the underlying models requires that the user regenerates the GMFGen model. Therefore, any customizations to the GMFGen model are lost and need to be re-programmed. Since model revisions occur more than often during the development process, going over the same process again and again is less than ideal and can lead to inconsistencies [71]. GTP addresses this issue, as the user can store model preferences in the M2M templates; these will be applied to the final GMFGen model before the code is regenerated.

To automate the process of generating the transformed GMFGen model, GTP adds a toolbar action to the Eclipse IDE. When called, the action will delete the old GMFGen model before generating a new one, it will then apply the M2M transformation to deliver the new GMFGen model and it will regenerate all Java code after it deletes any old code. This contribution, along with the M2M transformation mechanism, makes GTP a worth-having arrow to a GMF developer's quiver. GTP also deals with some of the GMF limitations, that we will discuss in chapter 6.

5.4. Eclipse Rich Client Platform

GMF is based on the Eclipse development platform [73]. Therefore, a typical scenario to run a GMF editor would prescribe to export the source code as an Eclipse plug-in, add it to the Eclipse IDE's plug-in list and use the editor through the Eclipse platform. This is sufficient for developers and people using the Eclipse environment as an open tools platform –i.e. to perform everyday tasks like reporting, modeling etc.-. As long as SNAPT targets not only to Business Process engineers, but to economists and business analysts as well, a big share of SNAPT's potential users is expected to be unfamiliar with the Eclipse environment. Moreover, we would expect SNAPT to have the look & feel of a standalone Service Network platform, rather than being an open tools platform facilitating the needs of Service Network modeling. Thus we have chosen to develop SNAPT as a standalone product, using GMF's Rich Client Platform (RCP) capabilities.

Although the Eclipse platform was originally intended to serve as an open tools platform, it is architected in such a way that its main components could be used to build any standalone client application. The minimal set of plug-ins required to build such a

rich client application is collectively known as the *Rich Client Platform (RCP)* [74]. Applications other than Integrated Development Environments can be built using a subset of Eclipse. These rich applications maintain Eclipse's dynamic plug-in model, while the user interface is built using the same toolkits and extension points. When building RCP applications, the layout and function of the workbench is under fine-grained control of the plug-in developer. A typical RCP application requires only two Eclipse plug-ins -the org.eclipse.ui plug-in and org.eclipse.core.runtime plug-in- as well as their prerequisites. As their names indicate, these plug-ins offer the application's runtime and user interface components respectively. RCP applications are also free to use any API necessary for their feature set and can require any plug-ins above the bare minimum, as expected.

In order to understand RCP applications and their architecture, one should first examine Eclipse's own architecture. As of release 3.0, Eclipse's core engine was redesigned to leverage RCP application development [75]. To accomplish this mission, the development team adopted an open standard provided by the OSGI Alliance[4], called the *OSGi Service Platform* [76]. The OSGi Service Platform consists of a component and service model as well as a management agent model. The first is used to define *OSGi components*, or *bundles*. A bundle is a set of Java packages that would be traditionally packaged in a JAR (Java archive) file. However, the bundle also contains dependency records –expressed as metadata-, that is, other Java packages it depends on. As a bundle needs to import packages not defined within its scope, it can also export internal packages used by other bundles. The OSGi Service Platform is responsible for automatically matching a bundle's imports and exports. When exporters are available for all packages that a bundle imports, we say that the bundle is *resolved*. Consequently, a resolved bundle can then later on be used for other bundles to be resolved. The OSGi Service Platform provides a service-oriented architecture enabling bundles to be activated. When a bundle is resolved, it may be started and stopped. The start enables a bundle's code to publish, find, and bind services (an OSGi service is a plain Java object). Started services, can be found for use by other services within a service registry; the bundle is responsible for registering created services. Please also note that services may be registered or unregistered at any time by an active bundle. The other architectural component of the OSGi Service Platform is the management agent model. This model is responsible for installing and uninstalling bundles, managing the configuration –i.e. starting and stopping bundles-, setting the permissions for bundles –e.g. importing and exporting packages permissions, service search permissions etc.- as well as ensuring the

[4] http://www.osgi.org/Main/HomePage

correctness of name equivalence –i.e. the right packages are used for resolving a bundle's dependencies-.

5.4.1. Rich Client Platform Architecture

The overall Eclipse architecture is shown in figure 13. On top of the Java Runtime Environment lie the OSGi Service Platform and Standard Widget Toolkit (SWT) layers. SWT is an open-source graphical java library that provides rich user interface components and functionality [77]. JFace is a convenient API on top of SWT providing helper classes for developing user interface features that can be tedious to implement [78]. All Eclipse and RCP dialogs and wizards are implemented using the JFace layer. The Workbench constitutes the application layer in which all other user interface components are displayed. On top of the workbench or directly onto the OSGi Service Platform can lie RCP specific plug-ins implementing the application's functionality.

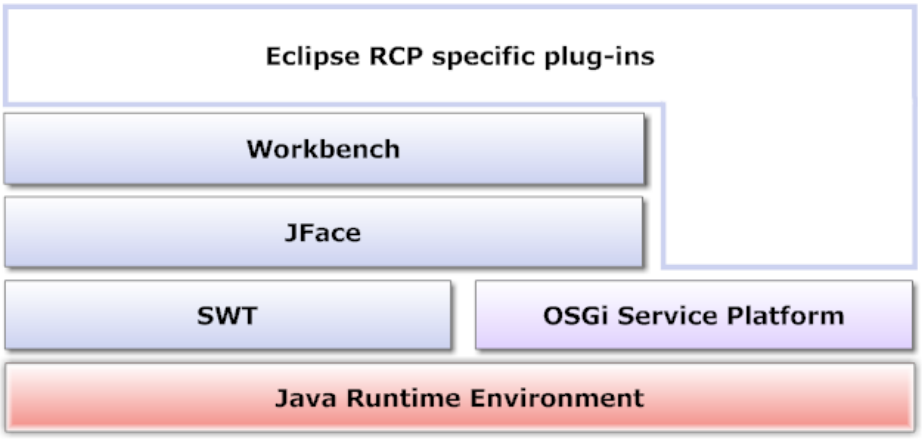

Figure 13. The overall RCP application architecture

5.4.2. The workbench

Once a GMF editor is generated using the Eclipse RCP, the workbench is also constructed. A screenshot of the workbench is depicted in figure 14. The workbench consists of five basic elements. The main menu (1), as with any typical application, allows access to the application's functionality. The toolbar (2) enables quick calls to frequently used actions, that otherwise are available through the main menu. The GMF editor (3) is the main editing window where the modeling is conducted. The outline view (4) provides a graphical overview of the GMF editor's diagram and the properties view (5) enables editing of properties for selected elements within the editor.

In Eclipse terminology, an *editor* is the primary component of the workbench. Just as there are different types of documents, there are different types of editors [79]. When

selecting or creating a new document in the Eclipse IDE, Eclipse attempts to open the document using the most appropriate editor. If it's a simple text document -i.e. a.txt file-, the document will be opened using Eclipse's built-in text editor. If it's a Java source file –i.e. a .java file-, it will be opened using the Java editor, which has special features like the ability to check syntax as code is typed. Additional editors are also contributed by the means of Eclipse plug-ins (bundles). In alignment with this policy, RCP applications can also embody several editors to allow editing of different file types. In a standard GMF RCP application, the primary editor initially available is the diagram editor contributed by the GMF-generated code –depicted by number 3 in figure 14).

A *view* in general, is designed to support interaction with the information in the workbench. As both the Eclipse IDE and an RCP application are designed to be extensible, views can be designed to show nearly any kind of information. Most commonly used views of the Eclipse IDE are views of the file system, the console view, the search file view etc. The outline and property views available in a GMF-RCP application –depicted by numbers 4 and 5 respectively in figure 14- are standard Eclipse views, customized to interact with the GMF editor.

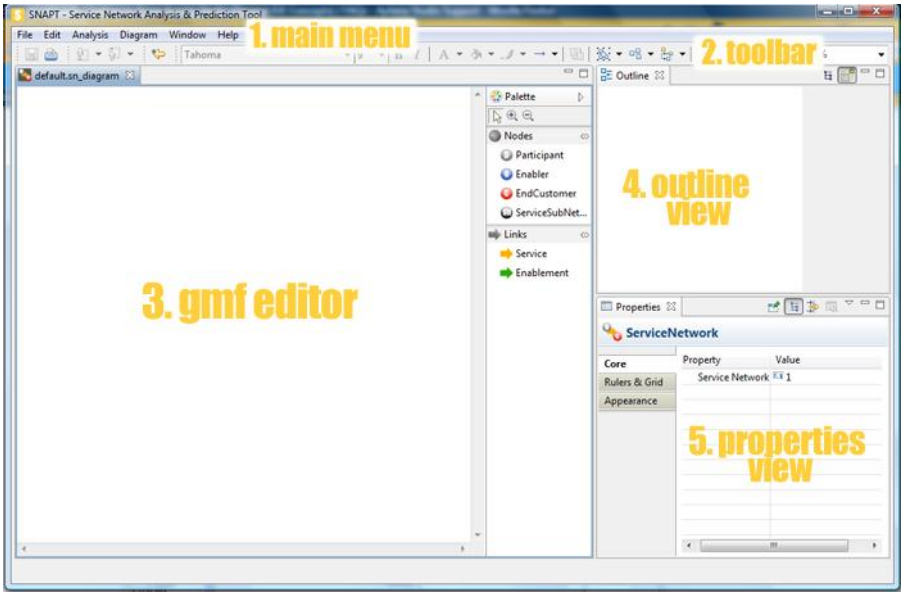

Figure 14. The GMF-RCP application workbench

Editors and views can be moved around practically anywhere in the workbench. A special layout of particular views and editors is called *perspective*. The Eclipse IDE has several perspectives facilitating the development of applications, debugging, plug-in development etc. The Eclipse user can easily switch between these perspectives to hide or show several views at once. An RCP application generated by GMF contains a single perspective, that is, the imaginable rectangle around the GMF editor and the outline and

properties views of figure 14. Within this perspective, the views and editors can be arranged in a convenient way to favor the user's workflow.

5.5. Service Network Analysis & Prediction Tool overview and architecture

Since SNAPT is intended to serve as a modeling & analysis tool for Service Networks, it should first of all be able to visualize both Service Networks and Service Sub-Networks following the methodology discussed in section 3.2. SNAPT should also enable assignment of KPOs for service providers and consumers; these are used to describe strategic goals and requirements. KPOs are like KPIs, only indicating an expected value instead of a measured value. A single KPI though, may apply to many services in concrete Service Networks –e.g. a KPI concerning availability may be used to describe up-time requirements for all web services within different Service Networks-. Therefore, KPIs should be defined outside the Service Network's scope; that way SNAPT users will not be forced to redefine a KPI each and every time they need to use it. Instead, they could be allowed to build their own KPI Library and later on assign KPOs to services using this library. As soon as a Service Network model is complete, SNAPT must also be in place to validate the model and provide feedback on errors and constraint violations. To facilitate Cost-Revenue analysis, the tool must also automatically calculate a Business Entity's costs and revenues based on its inputs and outputs. A Gordijn-style profitability sheet (see section 2.2.2) should also be exported in Excel format on a per Service Network basis, to provide better overview and control over costs and revenues. Finally, additional features such as the ability to extract a Service Network's semantic model –i.e. separate it from graphical information so that it can be imported to potential third-party applications- should also be supported.

To support Service Network modeling a single GMF editor hosted in an RCP application is required. Since we also need to model Service Sub-Networks that include SSN Input / Output ports not found in Service Network editor, a separate GMF editor must be implemented as well. The latter should interact with the Service Network editor, as Service Sub-Networks exist only in the context of a top-level Service Network. To facilitate the KPI library functionality, a third (not necessarily GMF-generated) editor must be developed. A custom view is also needed to enable users to assign KPOs to the Service Network model using the KPI Library. Last but not least, a set of UI dialogs, import and export wizards along with GMF modifications are required to support the remaining functionality.

The resulting top-level SNAPT architecture is shown in figure 15. As with any RCP application, the Java Run-time environment, OSGi Service Platform, SWT, JFace and

workbench layers constitute the basis of the application. The Service Network model as well as the KPI model –the latter used to define the KPI Library- are on top of the OSGi Service Platform as both are workbench-independent. All SNAPT core functionality –i.e. the three editors, the KPOs view, supplementary code and the additional graphical components- are implemented on top of the workbench and the two core models. Future extensions are also supported; therefore extra SNAPT specific plug-ins can be developed to add new functionality. These plug-ins could depend on SNAPT's core functionality, or plug directly onto the workbench or OSGi Service Platform layers.

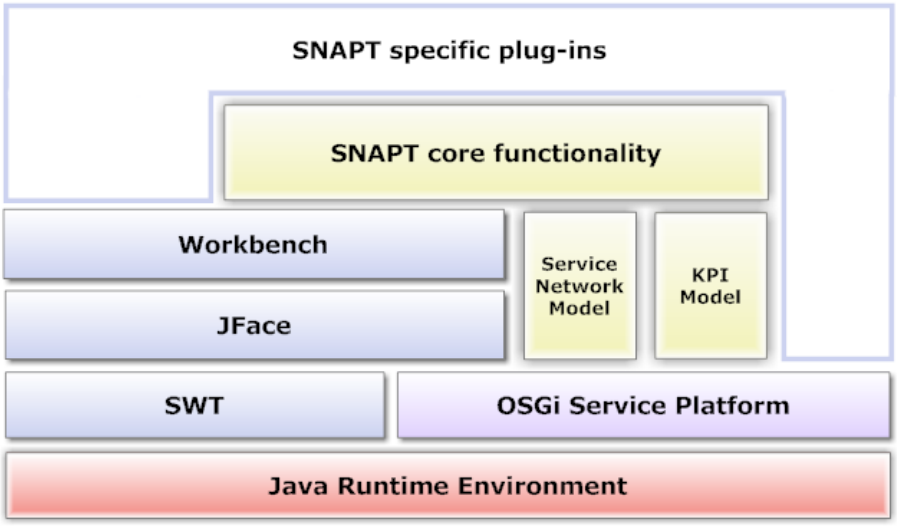

Figure 15. Top-level SNAPT architecture

6. Implementing the Service Network Analysis & Prediction Tool

In this chapter we discuss the development process followed to implement SNAPT. We present the Ecore models, editors, views and supplementary components that comprise SNAPT, from a technical aspect. Where needed, SNAPT screenshots and Java code snippets are also provided.

6.1. GMF editor models

SNAPT needs two GMF editors. The first will be used to model Service Networks (*SN editor*) and the latter to model Service Sub-Networks (*SSN editor*). Typically, this would require two completely independent projects, with discrete semantic and graphical models. However, SNAPT's editors have similar functionality, the only difference being a Service Sub-Network must contain input / output ports and links, which are not present in a Service Network. One could say that the SN editor is a subset of the SSN editor. Ideally, we would like both editors to use the same Java classes as far as the semantic (EMF) model is concerned, to avoid duplicate code. For example, a Participant object in the SN editor should instantiate the same class that a Participant in the SSN editor does. To achieve this we use a single Ecore model and a single generation model to generate EMF's Java code, and separate the editors on the GMF level –i.e. distinct GMFGraph, GMFTool, GMF Map and GMFGen models for each editor-. Within Eclipse, we start by creating a new project named org.eclipse.gmf.snapt and then create the Service Network Ecore model.

6.1.1. The Service Network Ecore model

The Service Network Ecore model (SNE model) is the core of the system. The SNE model is closely related to the Service Network meta-model discussed in section 3.1. In MDA terms, the Service Network meta-model is an abstract computational-independent model describing Service Networks, and the SNE model is a platform-independent model describing Service Network notation. To design the SNE model we have used the Ecore editor provided by EMF. The SNE model is shown as a UML diagram – automatically generated by the Ecore editor- in figure 16.

Service Networks consist of Business Entities and Services such as in the Service Network meta-model. However, in Service Network notation, service offerings, consumptions as well as the actual service are depicted by a double arrow. This double arrow is represented by the Service class in the SNE model. Each arrow has exactly one source and a target Business Entity. KPOs are also attached to the Service class, to represent both source and target entity's business goals. Please note that the KPO's "*type*" attribute is an enumeration declaring the type of the "*value*" attribute –i.e. a KPO's value may be a number, duration or unspecified-.

The SNE model defines the input and output ports of a Service Sub-Network as discussed in section 3.2 (SSNInputPort and SSNOutputPort classes respectively). Both port classes have a name corresponding to the name of the service they are associated with. The links connecting ports with Business Entities are represented by the SSNInput and SSNOutput classes. The links are defined in the scope of a Service Sub-Network just like their respective ports.

The SNE model is different from the Service Network meta-model as far as the enablement services are concerned. On the meta-model level, enablement services are just like any services offered and consumed in the network. Nevertheless, in Service Network notation enablement services target to services instead of a business entities (see section 3.2) and therefore need to be symbolized by a new class in the SNE model - the ServiceEnablement class. Enablement services have the same attributes as Services do.

Another difference between the two models is the Role class. Since in Service Network notation roles or names are just a label on a Business Entity node, and as there is currently no SNAPT functionality associated to the role an entity plays in the network (a future challenge concerning the use of roles in SNAPT is discussed in chapter 7), roles are embedded to the model as an attribute of the BusinessEntity class..

The SNE model also designates the attributes used as unique identifiers within a class and determines whether an attribute is unsettable–i.e. its value space includes the state of not being set- or not. This information along with the specified cardinalities is used by the validation component to decide the validity of a model instance.

All SNE model's classes must be defined in the context of a package. We name this *sn*, and set *http://tsl.gr/snapt/sn* as the namespace URI of the package.

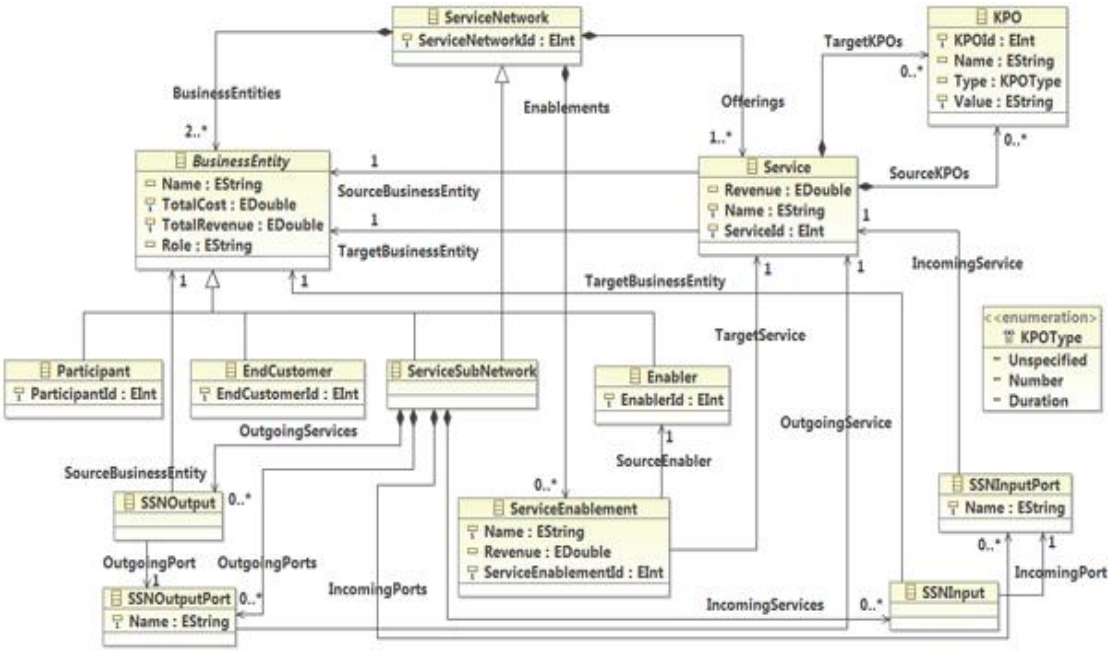

Figure 16. The Service Network Ecore model as a UML diagram

6.1.2. The generation model

An Eclipse wizard is provided to generate the GenModel, using the SNE model as input. The wizard automatically creates an elementary GenModel that should be modified before we proceed with code generation. We first need to set the base package for the Java code. In our case, this is the *org.eclipse.gmf.snapt* package. As a result, the model code will be placed in the *org.eclipse.gmf.snapt.sn, org.eclipse.gmf.snapt.sn.impl* and *org.eclipse.gmf.snapt.sn.util* packages –*sn* is the SNE model's package name-. The adapter code (ItemProviders, see section 5.2.2) will be placed under the *org.eclipse.gmf.snapt.sn.provider* package in a new, automatically created plug-in project (bundle) named org.eclipse.gmf.snapt.edit. The GenModel can also generate a basic model editor, but this is not needed since we plan to create GMF editors for the SNE model.

EMF supports model extensibility. One can define a new, independent Ecore model, create its GenModel and declare it to extend the original Ecore model. The new code is then perceived to be part of the original code and no further modifications are required. To enable this kind of extensibility to our model, we just set GenModel's "*Extensible Provider Factory*" attribute to "*true*".

Finally, we need to define which of the SNE model's class attributes will be user editable and which read-only. In general, we have set all attributes to be editable, except from unique identifiers (we want SNAPT to automatically generate IDs for the elements that users cannot modify) and the "*TotalCost*" and "*TotalRevenues*" attributes of the

BusinessEntity class (these should also be automatically calculated based on service revenues). As soon as the model is ready, we generate the model and adapter code.

6.1.3. The Service Sub-Network graphical model

The GMFGraph model is the first GMF-specific model, therefore we need an instance of this model for the SN editor and one for the SSN editor. Since the first is a subset of the latter, in this section we will describe the GMFGraph model for the SSN editor. The SN editor's GMFGraph model is a duplicate of the presented model, with all information regarding input / output ports and links removed.

In the GMFGraph model we define which classes are the nodes of our diagram and which are the connections. In alignment with the Service Network notation discussed in section 3.2 we set all *Participant*, *Enabler*, *End Customer* and *Service Sub-Network* classes to be nodes, and *Service* as well as *ServiceEnablement* to be the connections linking them. The *SSNInputPort* and *SSNOutputPort* classes are also nodes in the diagram, and *SSNInput* along with *SSNOutput* the connections linking ports with Business Entities. Please note that the *BusinessEntity* class is not a node; it only serves as a parent for the classes extending it, which in fact are the actual nodes. KPOs are also not represented on the Service Network notation. They are attached to services, and to allow KPO editing SNAPT will use a custom view. For each node class we want to create labels, so that the "*name*" attribute is printed on the canvas (figure 3). We also create labels for Services and ServiceEnablements, but not for SSNInputs and SSNOutputs as the name of the referenced service is displayed on the corresponding ports' label (figure 6).

For all nodes and connections we need to define their respective figures, that will represent the semantic elements on the canvas. For Participants, Enablers, End Customers and Service Sub-Networks we create four new figures, each of which contains an "*ellipse*" shape definition. Shapes, are predefined elements that set the shape of nodes and connections in the diagram –e.g. rectangle, ellipse, line, polyline etc.-. Since all four Business Entity classes have corresponding labels, we also need to add a grid layout to align them in the shape. The labels' graphical definitions are provided within each shape as well. We set the label's default text to be "<…>" –i.e. this is what will be displayed on the canvas if a Business Entity's name is empty- and also add a grid data layout to center the labels within the shape. As label *graphical* definitions are declared in the scope of their figure's shape, we need to create "Child access" elements, so that label definitions can access their respective graphical definitions within a figure. To demonstrate this, an example of the resulting XML code defining the figure, node and label elements for the EndCustomer class is provided below:

```
<figures name="Default">
    ...
    <descriptors name="EndCustomerFigure">
       <actualFigure xsi:type="gmfgraph:Ellipse"
           name="EndCustomerFigure">
          <layout xsi:type="gmfgraph:GridLayout"/>
          <preferredSize dx="74" dy="52"/>
          <children xsi:type="gmfgraph:Label"
              name="EndCustomerNameFigure" text="&lt;...>">
              <layoutData xsi:type="gmfgraph:GridLayoutData"
                  grabExcessHorizontalSpace="true"
                  grabExcessVerticalSpace="true"/>
          </children>
       </actualFigure>
       <accessors
  figure="//@figures.0/@descriptors.2/@actualFigure/@children.0"/>
    </descriptors>
    ...
</figures>
<nodes name="EndCustomer" figure="EndCustomerFigure"/>
<labels name="EndCustomerName" figure="EndCustomerFigure"
       accessor="//@figures.0/@descriptors.2/@accessors.0"/>
```

The same procedure is followed for the SSNInputPorts and SSNOutputPorts, the only difference being ports have a *"rounded rectangle"* shape. The SSNInput and SSNOutput connections must be depicted by a single directed arrow (figure 6). To achieve this we define a "polyline shape" and a "polygon decoration". The polyline shape represents a simple line connecting two nodes, and the decoration adds an arrow's head to the end of the line. ServiceEnablements are also represented by a polyline, but this time we set the line to be dashed, green-colored and we do not define a decoration (figure 4).

The Service connection should be represented by a double arrow (figure 3). Unfortunately, GMF does not support such a shape. Therefore, a custom Java class that creates a double arrow must be implemented. The custom class extends the polyline shape, and calculates the points of a dashed parallel line, added to the right of the original polyline. The new class is then set to be the shape of Services. The prototype of the Java class is shown below, followed by the resulting XML for the service figure descriptor:

In DoublePolyline.java

```
package org.eclipse.gmf.snapt.diagram.figures;
public class DoublePolyline extends PolylineConnectionEx {
   ...
}
```

In GMFGraph model file

```
<descriptors name="ServiceFigure">
    <actualFigure xsi:type="gmfgraph:CustomConnection"
           name="ServiceFigure"
```

```
4.              qualifiedClassName="org.eclipse.gmf.snapt.diagram.
5.              figures.DoublePolyline">
6.        <children xsi:type="gmfgraph:Label" name="ServiceNameFigure"
7.              text="&lt;...>"/>
8.     </actualFigure>
9.        <accessors
10.   figure="//@figures.0/@descriptors.4/@actualFigure/@children.0"/>
11.</descriptors>
```

6.1.4. The Service Sub-Network tools model

The GMFTool model defines the palette tools used to create graphical elements in the diagram. An example of a palette is shown in figure 14, to the right of the GMF editor depicted by number 3. In SNAPT, we would like to group the creation tools to enable quick and convenient access within the palette. The separation of node and link elements is a reasonable choice, so we create two different "tool groups": nodes and links. In the *nodes* group, we add creation tools for Participants, Enablers, End Customers and Service Sub-Networks, and set a name, description and appropriate images for the tools. Please note that we do not define tools for input and output ports, since these will be automatically created as soon as a service is linked to a Service Sub-Network. In the nodes group, we create tools for Services, Enablement Services, SSN Inputs and SSN Outputs. The GMFTool model for the SN editor is a duplicate of this model, the difference being that SSN Input and SSN Output tools are not defined for the SN editor.

6.1.5. The Service Sub-Network mapping model

In this model we map the nodes and connections defined in the GMFGraph model with tools defined in the GMFTool model. As with GMFGraph and GMFTool models, the GMFMap model must be created for both SN and SSN editors. In this section we describe the Service Sub-Network model. We first define the mappings for all diagram nodes: Participants, End Customers, Enablers, Service Sub-Networks, SSN Input ports and SSN Output Ports. The XML code setting the mapping for Service Sub-Networks is shown below:

```
1. <nodes>
2.    <containmentFeature
3.         href="SNAPT.ecore#//ServiceNetwork/BusinessEntities"/>
4.    <ownedChild relatedDiagrams="//@diagram">
5.       <domainMetaElement href="SNAPT.ecore#//ServiceSubNetwork"/>
6.       <domainInitializer xsi:type="gmfmap:FeatureSeqInitializer">
7.          <initializers xsi:type="gmfmap:FeatureValueSpec">
8.             <feature xsi:type="ecore:EAttribute"
9.                  href="SNAPT.ecore#//BusinessEntity/Name"/>
10.             <value body="//" language="java"/>
11.          </initializers>
12.          <initializers xsi:type="gmfmap:FeatureValueSpec">
13.             <feature xsi:type="ecore:EAttribute"
14.            href="SNAPT.ecore#//ServiceNetwork/ServiceNetworkId"/>
```

```
15.              <value body="//" language="java"/>
16.           </initializers>
17.        </domainInitializer>
18.        <labelMappings xsi:type="gmfmap:FeatureLabelMapping"
19.             readOnly="true" viewPattern="{0} OR {1}">
20.          <diagramLabel href="SSN.gmfgraph#ServiceSubNetworkName"/>
21.           <features href="SNAPT.ecore#//BusinessEntity/Name"/>
22.           <features href="SNAPT.ecore#//BusinessEntity/Role"/>
23.        </labelMappings>
24.        <tool xsi:type="gmftool:CreationTool"
25.             href="SSN.gmftool#//@palette/@tools.0/@tools.3"/>
26.        <diagramNode href="SSN.gmfgraph#ServiceSubNetwork"/>
27.     </ownedChild>
28. </nodes>
```

The "*nodes*" element is used to describe a node mapping, as its name suggests. The containment feature sets the node's parent element; in the Service Sub-Network case, a newly created node belongs to a Service Network's Business Entities set. The "*ownedChild*" tag is used to define the actual mapping. Service Sub-Networks are special nodes, as we want to allow editing of their internal structure: When double-clicking on a Service Sub-Network's graphical element, an SSN editor should open in a new window. To enable this functionality we set the "relatedDiagrams" to reference the SSN diagram editor itself. The semantic class defining Service Sub-Networks is the located in the SNE model, and is referenced by the "*domainMetaElement*" tag. After we set the semantic element we need to provide the node's "*Feature initializers*". Feature initializers, in general, are used to describe actions that will take place whenever a new element is created. When a new Service Sub-Network is created, we want SNAPT to automatically generate a unique identifier and a default name for the Service Sub-Network. To accomplish this, we use the code between the 6th and 17th line of the above snippet. We reference the *Name* and *ServiceNetworkId* attributes (Service Sub-Networks inherit their id from the parent *ServiceNetwork* class) and tell GMF to generate appropriate java methods, that we will later on explicitly code to achieve the desired functionality. We now need to set the mappings for the Service Sub-Network label, so we reference the respective label defined in the GMFGraph model. When a Service Sub-Network's name is not set or is empty (SNAPT users are able to delete automatically generated names), we want SNAPT to display the Service Sub-Network's role instead. This "name or role" pattern is set through the "viewPattern" attribute, where name is the first and role the second of the referenced class attributes. Please note that the pattern expression uses Java's built-in Message Format class[5], which does not support the "*OR*" keyword by default. To achieve the desired effect we needed to modify the expression parser on the GMF level, after the code has been generated. Since the label can refer to both the name and role of the Service Sub-Network, we do not allow users to edit the label directly

[5] http://java.sun.com/j2se/1.5.0/docs/api/java/text/MessageFormat.html

from the canvas; they are forced to use the properties view (shown in figure 14). To accomplish this we just set the "*readOnly*" attribute to "true". Finally, to complete the Service Sub-Network mapping, we just reference the respective creation tool -defined in the GMFTool model- and the Service Sub-Network node element of the GMFGraph model, so that GMF will be able to map them to the semantic element.

We follow the same procedure for all remaining diagram nodes. The mapping code for diagram connections is also similar. However, in the case of connections, we also need to set a few constraints: We want to inhibit a Service to be consumed by the Business Entity that offers it. At the same time, we should not allow an Enablement Service to enable a third Service offered or consumed by its Enabler. Finally, a Service cannot be offered by an End Customer. These kind of constraints are defined in the GMFMap model. To describe them, we have used the OCL language [80], as GMF by default supports OCL expressions *on the diagram level*. Nevertheless, it is not possible to set OCL constraints into the Ecore model (semantic level), which is the appropriate layer for such restrictions –e.g. the fact that a service cannot be consumed by its provider is a semantic rather than a graphical constraint-. This constraint problem is more of EMF's responsibility, and since GMF's workaround is sufficient, we define our restrictions in the GMFMap model.

All the above constraints are used to prevent undesired user actions in the SSN editor. When a user attempts to create a new Service and link both of its edges to the same BusinessEntity, SNAPT will not allow the creation of the Service at the first place. However, there exist constraints we should check when validating a model rather than when creating elements. Such constraints can also be defined in the GMFMap model; they are called "*Audit rules*". We have defined two audit rules –again using the OCL language-: The first is used to check if a Business Entity has at least one of its Name or Role attributes set. The latter, is called to ensure that all Business Entities in the network offer or consume at least one Service –i.e. there are no "orphan" nodes-. To demonstrate an OCL expression, the code for this constraint is provided below. The code iterates over all Service Network's Services and Enablement Services to find at least one Service referencing a given Business Entity. When validating the model, the OCL expression is called for all Business Entities within the Service Network.

```
1. ServiceNetwork.allInstances()->forAll( sn : ServiceNetwork
2.     | if sn.BusinessEntities->includes(self) then
3.         (sn.Offerings->collect(o : Service |
4.             o.SourceBusinessEntity)->asSet())->includes(self) or
5.         (sn.Offerings->collect(o : Service |
6.             o.TargetBusinessEntity)->asSet())->includes(self) or
7.         (sn.Enablements->collect(se : ServiceEnablement |
8.             se.SourceEnabler)->asSet())->includes(
9.                     self.oclAsType(Enabler))
10.      else true
11.      endif
12.  )
```

6.1.6. The generator models

An elementary GMFGen model is automatically generated using all GMFGen, GMFGraph and GMFTool models as input. Using an Eclipse wizard we generate the model of the SN and SSN editors. These have to be readjusted to facilitate SNAPT's needs before we proceed with code generation.

Since both GMFGen models derive from the same Eclipse modeling project, GMF automatically generates the same "*PluginID"* and "*EditorID*" for both editors. However, for the OSGi Service Platform to be able to correctly resolve the dependencies, these attributes must have different values. As we explicitly set the ID of the SSN editor, we also need to set the "*openDiagramBehavior*" model attribute to reference this ID; that way both editors will know what to do when double-clicking on a Service Sub-Network.

By default, GMF uses two separate files for storing semantic and diagram information. Since this is not a common practice in other applications, it could be tricky for inexperienced users that could accidentally delete or move one of the two files, leading to inconsistencies. To avoid this, we set the "*sameFileForDiagramAndModel*" attribute to "true", and later on implemented a wizard enabling the export of a Service Network's semantic model to a separate file. We also set the file extension for Service Network models to be "*sn_diagram*", and for Service Sub-Network files "ssn_diagram". However, note that we will not allow SSN editors to open and save separate Service Sub-Network files. An SSN editor will only enable editing of Service Sub-Networks contained in a top-level Service Network. Therefore, SNAPT will deal only with ".sn_diagram" files.

Since a set of custom Java classes have been implemented to supplement GMF's functionality –e.g. the DoublePolyline class-, and as we would rather not mix custom code with the automatically GMF-generated code, we created a new Eclipse plug-in, named *org.eclipse.gmf.snapt.prerequisites,* that contains the custom code. This package must be automatically imported to all GMF-generated Java plug-ins, so we add it to the GMFGen model's "*requiredPlugins*" list, along with the GTP packages.

A few modifications are also required to enable the printing and validation of models, which are by default disabled. To store all these changes, we use the GTP template extension mechanism. A fragment of the template for the SSN editor is provided below, to demonstrate the approach:

```
1. internalTransform(GenPlugin this) :
2.     setPrintingEnabled(true)->
3.     setID("org.eclipse.gmf.snapt.ssn.diagram")->
4.     requiredPlugins.add("de.itemis.gmf.runtime.extensions")->
5.     requiredPlugins.add("org.eclipse.gmf.snapt.prerequisites");
```

The mechanism simply iterates over the GMFGen model's elements, and as soon as it finds a match, it performs the operations suggested by the template. As a result, two transformed GMFGen models are created and we can now generate the Java code.

6.2. Sharing an editing domain

The editing domain of a GMF editor is responsible for holding all model resources in the memory, managing transactions –i.e. executing user commands and handling undo / redo context-, storing adapter factories, etc. Normally, the GMF generated code creates a new editing domain each time a new diagram editor is opened. This can lead to several problems when using different editors to edit a single model:

1. Multiple instances of the same model elements are held in memory at the same time, resulting in large memory overhead.

2. Editors synchronize their contents only on file changes. If a file resource change triggers a reload in a dirty editor –i.e. an editor whose resource is modified by the editor, but the changes are not yet saved; a dirty editor is depicted by a star to the left of the editor's tab, as shown in figure 17-, the user has the choice to keep either the changed version in the editor or the currently saved one. This results in information loss and insufficient synchronization loops.

Especially the latter, is a problem that SNAPT needs to resolve since new SSN editors open every time a user double-clicks on a Service Sub-Network element. To address this issue we use GTP, that provides a mechanism for creating a shared editing domain. Using such a mechanism has a number of consequences to SNAPT's behavior, some which are welcome while others are not.

Except from the editing domain, editors also share their command stack. In Eclipse, each editor or view has its own command stack; the undo / redo list changes each time a different UI component is activated. By sharing the command stack undo / redo affects all other editors than the active one. Nevertheless, this is the desired behavior in SNAPT,

which is an RCP application and does not have to follow the Eclipse IDE undo / redo policy: using several command stacks would possibly frustrate users who are unfamiliar with Eclipse's environment.

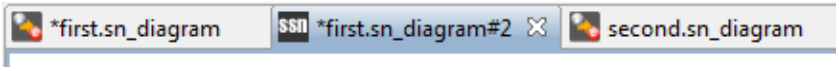

Figure 17. SNAPT editor bar screenshot. Editors using the same model should all get dirty on model changes

Editors sharing the same editing domain are also either all dirty or all clean. Let's suppose we open a Service Network diagram named "first.sn_diagram", from the file system using SNAPT. If we double-click on a Service Sub-Network of the diagram a new SSN editor opens. Finally, let's assume we also load a second Service Network model from the file system, named "second.sn_diagram". When adding a new Participant in the SSN editor, the change should affect the SN editor holding the "first.sn_diagram" resource as well. Both editors should be marked dirty, but the SN editor used to display the "second.sn_diagram" must not be affected. This is demonstrated in figure 17. Since all editors sharing an editing domain are all dirty or all clean, we need to interfere with GTP and customize its code to achieve the desired functionality. We have implemented an iteration over all open editors that use the shared editing domain, to determine whether they reference the changed model or not. The editors referencing the model are then marked as dirty.

When using a shared editing domain, the domain should no longer be disposed when an editor is closed, since other editors may be using it. Nevertheless, an unloading strategy is needed to do away with unneeded resources and avoid memory leaks. This strategy is already implemented in GTP. When unloading resources though, references of the unloaded models might still exist in the command stack. A cleanup strategy is not developed yet [81], and this is the only undesired consequence when using GTP's shared editing domain. To avoid inconsistencies, the SNAPT command stack resets each time a model is unloaded.

To use the shared editing domain, GTP requires that we make some modifications to the GMFGen models and then regenerate the code. First of all, distinct model and plug-in IDs, as well as distinct diagram file extensions are required. We have already manually set these attributes for our own reasons, so we change nothing here. The "*dynamicTemplates*" attribute must also be set to "true", and the template directory should point to the GTP sharedEditingDomain template directory. That way, GTP code modifications will be automatically applied to the GMF-generated code. Additionally, distinct visual IDs must be set for the diagram editors and all nodes and links. Visual IDs

are automatically regenerated every time a GMFMap to GMFGen transformation occurs. To preserve our manually defined IDs upon retransformation, we use GMF's org.eclipse.gmf.bridge.trace plug-in. This plug-in will create an additional trace file during GMFMap to GMFGen transformation and will use it to dispatch visual IDs on next transformation. Once the trace files were created, we manually modified them to hold our own visual IDs . Finally, GMF provides an *ElementTypeRegistry*, where all diagram elements are registered. In SNAPT, SN and SSN editors use more or less the same semantic elements; these should be registered only once in the ElementTypeRegistry. This is accomplished using GTP's toolbar action described in section 5.3. The action searches for potential duplicate entries and replaces them with references to the already registered element.

6.3. Implementing clipboard actions

One of GMF's weak points is the copy / paste functionality. Although clipboard actions are implemented, they do not behave as expect, or at least do not facilitate SNAPT's needs. Whenever an element is copied, it is the graphical component that only gets copied and not its underlying semantic element. The new graphical element just references the same semantic element that the original element does. For example, if we attempt to create a copy of an End Customer, we only get a new ellipse shape on the diagram pointing to the very same End Customer. If we then delete the new End Customer, both End Customers disappear as the semantic element gets deleted as well. Nevertheless, as an addition to the copy command, a *duplicate* function is provided by GMF. Although this command does copy the underlying semantic element, it does not generate a new ID for the duplicate element. The element IDs are SNAPT's responsibility; whenever a new element is created SNAPT automatically calculates a unique identifier. We need to embed this functionality on copy actions as well. For all the above reasons, we must create our own copy / paste functionality that will override GMF's default actions. GMF's original cut action behaved as expected but was also re-implemented to allow moving elements to different editor types. As a result, an element can now be cut from an SN editor and be pasted to an SSN editor, which was not possible in the first place.

Clipboard functionality is controlled by an entire mechanism being responsible for managing global actions. To allow replace of the copy / paste actions, we first need to create our own global action handler. We create a new Java class named "*ClipboardActionHandlerProvider*" and place it under a new package (org.eclipse.gmf.snapt.diagram.common.action.handlers) in our *org.eclipse.gmf.snapt.prerequisites* bundle. In the plugin.xml file of the bundle containing

our RCP application –i.e. a plugin.xml file is used to define plug-in extension points -, we explicitly set the new class as the handler for global actions, using the following code:

```
<extension point="org.eclipse.gmf.runtime.common.ui.services.
     action.globalActionHandlerProviders" id="global-actions">
   ...
   <GlobalActionHandlerProvider class="org.eclipse.gmf.snapt.
    diagram.common.action.handlers.ClipboardActionHandlerProvider"
      id="org.eclipse.gmf.snapt.snRender">
      ...
   </GlobalActionHandlerProvider>
   ...
</extension>
```

The action handler provider is responsible for creating the instance of the action handler which is responsible for returning the appropriate command. Our custom action handler extends GMF's original action handler, so that copy, cut and paste functionality can be overridden. That is, we must implement our own *canCopy*, *canCut*, *canPaste*, *getCommand*, *getCopyCommand*, *getCutCommand*, *getExecutePasteAfterCutCommand* and *getExecutePasteAfterCopyCommand* methods. All method names are self-explanatory, we should only remark that the getCommand method is a controller method responsible for returning executable commands for copy, cut or paste requests.

When pasting an element, the action handler needs to know in which of the new container's attributes lists the element will be appended. For example, when pasting an End Customer, it should be appended to a Service Network's or Service Sub-Network's BusinessEntities list. Since the action handler is a generic class dealing with all kinds of elements, it is the container's responsibility to inform the action handler where to append an element. To overcome this, we have created a new interface named *ViewAndFeatureResolver*. Both Service Network and Service Sub-Networks have their own implementations of this interface. When the action handler is requested to paste an item, it first obtains the container's ViewAndFeatureResolver class. Depending on the type of the element to be pasted, the ViewAndFeatureResolver class will then return the list to which the item will be appended. Classes like the ViewAndFeatureResolver are called "*adapters*", since they are used to adapt a Java class –e.g. the Service Network class- to facilitate the needs of the adapter's requester –e.g. the action handler class-; the use of adapters is a common practice in GMF. A short example of ViewAndFeatureResolver class's usage is provided below.

In ViewAndFeatureResolver.java

```
public interface ViewAndFeatureResolver extends ViewResolver {
     EStructuralFeature getEStructuralFeatureForEClass (EClass eClass);
}
```

In ServiceNetworkEditPart.java

```
1. private ViewAndFeatureResolver resolver = new ViewAndFeatureResolver(
2.    @Override
3.    public EStructuralFeature getEStructuralFeatureForEClass(EClass
4.                                                              eClass) {
5.       if(SnPackage.eINSTANCE.getBusinessEntity().isSuperTypeOf(eClass))
6.          return SnPackage.eINSTANCE.getServiceNetwork_BusinessEntities();
7.       if(SnPackage.eINSTANCE.getService().isSuperTypeOf(eClass))
8.           return SnPackage.eINSTANCE.getServiceNetwork_Offerings();
9.       if(SnPackage.eINSTANCE.getServiceEnablement().isSuperTypeOf(eClass))
10.        return SnPackage.eINSTANCE.getServiceNetwork_Enablements();
11.     return null;
12.   }
13.   ...
14. }
15. ...
16. @Override
17. public Object getAdapter(Class adapter) {
18.    if (adapter != null && adapter.equals(ViewAndFeatureResolver.class)){
19.       return this.resolver;
20.    }
21.    return super.getAdapter(adapter);
22. }
```

In ClipboardActionHandler.java

```
1.  @Override
2.  protected boolean canPaste(IGlobalActionContext cntxt) {
3.     ...
4.     EStructuralFeature feature = getFeature(firstObjectToBePasted,
5.                                                        pasteDestination);
6.     if(feature != null) {
7.        if(feature.getEType().getInstanceClass().
8.                                       isInstance(firstObjectToBePasted))
9.           return true;
10.    }
11.    ...
12. }
13. ...
14. protected EStructuralFeature getFeature(EObject element, EObject
15.                                    objectToBePasted, EditPart editPart) {
16.    ...
17.    EStructuralFeature feature = null;
18.    Object adapter = editPart.getAdapter(ViewAndFeatureResolver.class);
19.    ViewAndFeatureResolver resolver = null;
20.    if(adapter instanceof ViewAndFeatureResolver)
21.       resolver = (ViewAndFeatureResolver)adapter;
22.    ...
23.    //calculate feature
24.    return feature;
25. }
```

To complete the clipboard functionality, a set of additional classes is also developed. The majority of these classes is command wrappers and utility providers, and therefore we will not expand on their implementation.

6.4. Service Network and Service Sub-Network editor issues

While in the development process, we came across several additional issues. In some cases GMF did not behave as SNAPT required, while in others supplementary functionality should be implemented to support the targeted feature set. The final list of modifications and additions is not inconsiderable, therefore in this section we briefly describe an indicative set of the problems and their respective solutions.

6.4.1. Automatically generating unique identifiers and names

SNAPT needs to automatically generate unique identifiers along with universal default names whenever a new element is created on the canvas. In section 6.1.5, we described how feature initializers were added to the GMFMap models. Upon code generation, these initializers created a Java class named *ElementInitializers.* The ElementInitializers class contains methods that GMF calls when a new element is created and appropriate methods are generated for each element having initializers. The code calculating both IDs and names is injected in these methods. SNAPT element IDs are integers. To find the ID for an element we start from number 1 and increase by 1 each time the current number is occupied by another instance. The returned ID is the first not to be occupied. IDs are unique among the same type of elements. For example, both an End Customer and a Participant can have ID "1". A similar procedure is followed for resolving an element's name. The name attribute consists of the element's class name followed by an integer. The first Participant of a Service Network will be named "Participant1", the second "Participant2" etc. Please note that IDs and names are calculated on the Service Network level. Elements contained in Service Sub-Networks must be uniquely identified among other Service Network elements and not only among their siblings.

6.4.2. Cost – revenue calculations

SNAPT must also automatically calculate costs and revenues for Business Entities participating in a network. Each time a Service's revenue attribute value changes, a notification is generated. We have registered the Service's edit part class –i.e. the class holding information on a graphical element and its semantic elements- to listen to these notifications. As soon as the notification is received, the edit part updates the cost and revenue values of its source and target Business Entities as explained in section 3.3. Please note, that all modifications made to support calculations on Services, are also implemented to support Enablement Services. When a Service targets to a Service Sub-Network, the revenues that it produces must also be assigned as costs to the Business Entity consuming the Service *inside* the Service Sub-Network. The code responsible for

performing this task is embedded to the SSN Input connection element initializing method –located in the ElementInitializers class -. It is when a link connecting an SSN Input Port with a Business Entity is created, that the target Business Entity's costs must be updated. The new connection's edit part is also registered to listen to notifications generated by Services, so that whenever the Service's revenues change, the SSN Input's target Business Entity gets updated. The cost - revenue mechanism, must also account for many scenarios: a user may delete a Service, reorient a Service –i.e. reconnect it to another source or target Business Entity-, delete or reorient SSN Input links or Enablement Services etc. In GMF, before a command is executed, an intermediary class is called. This class is called edit helper. Edit helpers can be implemented for any diagram element. With edit helpers, additional code can be executed just before a command is added to the command stack. In our case, the *ServiceEditHelper* class may define the behavior of Services when they are deleted or reoriented. Using the ServiceEditHelper, ServiceEnablementEditHelper and SSNInputEditHelper classes, we have implemented a series of methods that handle all these scenarios. A sample code demonstrating calculations performed when reorienting an SSN Input is provided below:

```
1.  protected ICommand getDestroyDependentsCommand(
2.                                              DestroyDependentsRequest req) {
3.      ...
4.      final Service service = ((SSNInput)req.getElementToDestroy()).
5.                                  getIncomingPort().getIncomingService();
6.      final BusinessEntity target = ((SSNInput)req.getElementToDestroy()).
7.                                  getTargetBusinessEntity();
8.      ...
9.      try {
10.        //change cost for SSNInput's target business entity
11.        target.setTotalCost(target.getTotalCost() - service.getRevenue());
12.     } catch (Exception er) {
13.        er.printStackTrace();
14.     }
15.     ...
16. }
```

The getDestroyDependentsCommand method is called when an SSN Input is about to be deleted. Please note that the command framework is built in such a way, that the method is also called as a result of a destroy commands' chain. When a Service Sub-Network is deleted, its internal Business Entities, Services, Enablement Services, ports and inputs / outputs must also be destroyed. In this case, the framework makes subsequent calls to the destroy commands of each one of these elements.

6.4.3. Automatically creating SSN Input Ports and SSN Output Ports

Each time a Service is connected to a Service Sub-Network, SNAPT must automatically create input or output ports –depending on whether the Service Sub-Network is the

service provider or consumer- inside the Service Sub-Network. In the ElementInitializers class, we add a code snippet that gets called as soon as a Service is created. It first checks whether the source and target Business Entities are Service Sub-Networks and then creates the appropriate port elements if necessary. Corresponding snippets are also added in the ServiceEditHelper class, to delete and create ports whenever a Service is deleted or reoriented. SSN Input Ports and SSN Output Ports are named after the Service they reference. If the user changes a Service's name and this Service is connected to a Service Sub-Network, then the corresponding port's name label must also change. To accomplish this, we have registered the edit helpers of both SSN Input Port and SSN Output Port labels to listen to notifications generated by the Service semantic element, just like with cost - revenue calculations. As soon as the Service's name changes, a callback method in the edit helper class gets called, and updates the port's label. There might be cases, where an SSN Input Port is connected to a Service Sub-Network. In this case, a new input port must also be created in the second-level Service Sub-Network. This is to be checked whenever a new SSN Input link is created. Therefore, the ElementInitializers and SSNInputEditHelper classes must be enhanced to provide similar functionality with that of Services. Finally, input and output ports must not be created, deleted or modified in any way from the user interface. In our GMFTool model (see section 6.1.4) we did not define creation tools for SSN input and output ports, therefore these elements cannot be created from the palette. However, GMF has automatically generated additional pop-up creation tools, that show up whenever the mouse rolls over the canvas. These pop-up tools, include creation tools for input and output ports. To remove the tools, we had to modify a class named *SNAPTModelingAssistantProvider*, by simply commenting out the lines defining these tools. To prevent deleting input / output ports, we need to modify the SSNInputPortItemSemanticEditPolicy and SSNOutputPortItemSemanticEditiPolicy classes. In GMF, edit policies are used to define how an element will behave when an action is requested. In our case, we set the edit policies to forbid deleting the underlying the semantic elements as follows:

```
1. protected Command getDestroyElementCommand(DestroyElementRequest req) {
2.     return UnexecutableCommand.INSTANCE;
3. }
```

6.5. Wizards and user interface components

SNAPT allows users to export a Service Network's semantic model, export Service Sub-Networks to stand-alone Service Networks as well as import Service Network semantic models to initialize diagrams. The Eclipse IDE already incorporates an internal

mechanism for handling import and export wizards. Although not part of an automatically generated GMF-RCP application, this Eclipse mechanism can be used by any RCP application as long as it gets activated. Therefore, prior to implementing the wizards that perform all desired tasks, we need to enable import and export wizards in SNAPT. To do this, we add the following XML code in the plugin.xml file of our RCP application:

```
1.  <extension point="org.eclipse.ui.menus">
2.     <menuContribution locationURI="menu:file?after=export">
3.        <command commandId="org.eclipse.ui.file.export" style="push">
4.        </command>
5.     </menuContribution>
6.     <menuContribution locationURI="menu:file?after=import">
7.        <command commandId="org.eclipse.ui.file.import" style="push">
8.        </command>
9.     </menuContribution>
10. </extension>
```

The above code contributes two new elements to SNAPT's "File" menu: the import and export menu elements. When used, each one of these items will call a command that is implemented and defined in Eclipse's source code. This command opens a new window that displays all import or export wizards registered in SNAPT. Figure 18 shows SNAPT's export wizard window, as displayed after the "export" menu button is pressed. New wizards can be registered to these wizard pages through the "*org.eclipse.ui.importWizards*" and "*org.eclipse.ui.exportWizards*" extension points.

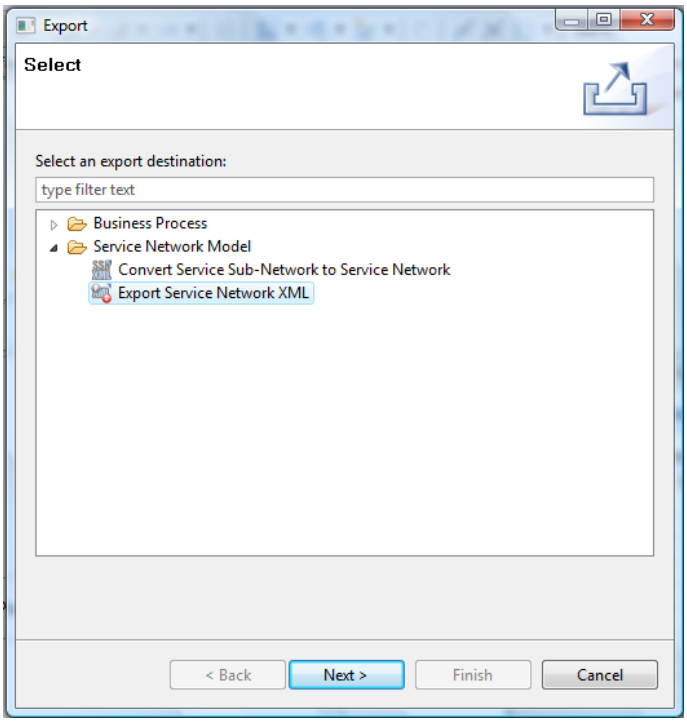

Figure 18. The SNAPT export wizards page

6.5.1. Export Service Network XML wizard

We first discuss the wizard used to export a diagram's semantic model to a standalone model file. This wizard is named "Export Service Network XML", as shown in figure 18. The wizard is registered to SNAPT's export wizards list using the "org.eclipse.ui.exportWizards" extension point. To implement it, we start by creating a class named *ExportSnToSnWizard* that extends the *Wizard* class and implements the *IExportWizard* interface. The classes or interfaces that an Eclipse-based wizard should extend or implement, are defined in the Eclipse framework. Wizards, in general, consist of *Wizard Pages.* SNAPT wizard pages are designed to be reusable. For example, a wizard page used to select a file path can be used in all export wizards. The "Export Service Network XML" wizard is a one-page wizard. The user just selects which of the open SN editors' diagram will be exported and sets the file path. The class implementing this wizard page is named "SelectResourceWizardPage" and extends the "SelectFilePathWizardPage" class. The latter creates a "Browse.." button on the wizard dialog where the user is able to choose a file path and the first extends it by adding a selection box that displays all available resources from where the user will choose. When the user clicks on the "Finish" button, a callback method that exports the selected semantic model is triggered.

6.5.2. Convert Service Sub-Network to Service Network wizard

Service Sub-Networks include input and output ports as well as input and output links. To convert Service Sub-Networks to standalone Service Networks, one should just replace ports and links with the Business Entities and Services that they reference. An example is shown in figure 19. The Service Network depicted by number 1, consists of a Service Sub-Network and two Participants. One of these Participants -named "Outside 2"- offers a Service to the Service Sub-Network. The internal structure of the Service Sub-Network is shown in the middle of figure 19. To create a new Service Network that derives from this Service Sub-Network, we replace the "Outside Service 2" SSN Input Port with the "Outside 2" Participant; we then use the "Outside Service 2" that was referenced by the input port to connect "Outside 2" and "Inside 1" Participants. The resulting Service Network is depicted by number 3 in figure 19. To allow conversion of Service Sub-Networks to Service Networks we create the "ExportSsnToSnWizard" class and register it to SNAPT's export wizards through the "org.eclipse.ui.exportWizards" extension point. This will also be a single page wizard, so we set it to extend the "ExportSnToSnWizard" class; that way only the needed methods should be overridden The new wizard page is named "SsnToSnWizardPage" and extends the

"SelectResourceWizardPage" by adding a new select box, where the user can choose the Service Sub-Network he wishes to export. When clicking on the "Finish" button, a code snippet resolves the Service Network and flushes it to the specified file.

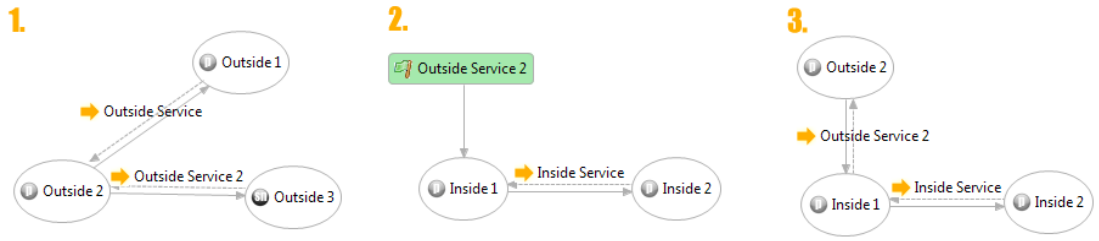

Figure 19. 1) A Service Network 2) The "Outside 3" Service Sub-Network's internal structure 3) The Service Sub-Network converted to a standalone Service Network

6.5.3. Import Service Network XML wizard

SNAPT must also allow the import of Service Network semantic model files and automatically initialize a diagram to represent them on the canvas. GMF does support this initialization by default, so we only need to adapt the original GMF action to cooperate with the export wizard mechanism. We have implemented a new wizard - named "ImportSnWizard"- that contains a single "SelectFilePathWizardPage" so that SNAPT users can select the file to import. On finish, we only call the GMF action being responsible for initializing the Service Network diagram. To register the wizard to SNAPT's set of import wizards we use the "org.eclipse.ui.importWizards" extension point.

6.5.4. Generating profitability sheets

Except from import and export wizards, a UI dialog that enables the export of profitability sheets has also been developed. SNAPT profitability sheets are Excel files that overview cost - revenue calculations within a Service Network. Separate Excel sheets are used to display each Business Entity's costs and revenues; an additional sheet is also used to summarize the costs / revenues for the entire network. Within the Excel file, formulas are assigned to appropriate cells, connecting Service revenues with Business Entities' total costs and total revenues. That way, when a Service revenue cell is modified, the changes automatically affect the costs and revenues of the Business Entities consuming and providing the Service respectively. To generate the Excel files, we have used the Apache POI project [82], which offers an API for manipulating Microsoft documents in Java applications.

6.6. The KPI editor and the KPOs view

The KPI editor is an EMF-generated editor used to define KPIs in SNAPT. We choose an EMF editor instead of implementing our own from scratch since EMF editors are effective, follow the Eclipse architecture and -most of all- are model-driven. To keep things simple, the KPI Editor is not allowed to create, load or save multiple KPI files. SNAPT treats the KPI editor the way traditional applications use front-ends for a single database. A single KPI Library file is used to store the KPIs defined by the users, and the KPI editor opens that file by default. To generate the KPI editor an Ecore model as well as a GenModel are required.

6.6.1. KPI editor models

The KPI Ecore model is shown in figure 20, as a UML diagram. Its structure is pretty simple. A KPI Library consists of categories, that contain sub-categories. KPI are contained within these sub-categories. Both *Category* and *Subcategory* classes have a name and a unique identifier as attributes. KPIs have an ID, a name and an optional description

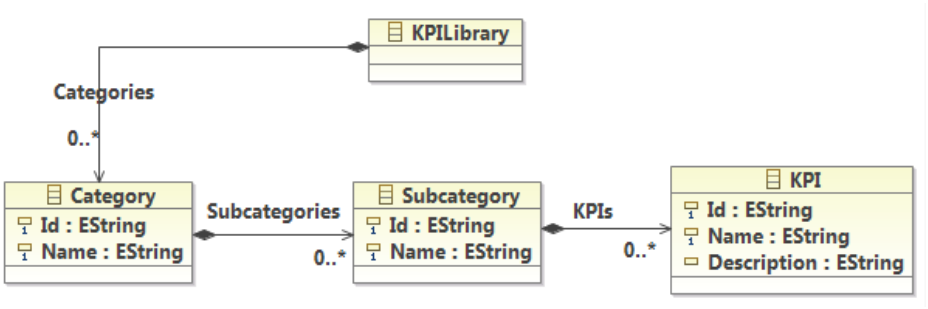

Figure 20. The KPI Editor Ecore model as UML diagram

The GenModel for the KPI editor is generated through an Eclipse wizard that uses the KPI Ecore model as input. We want the KPI Editor to allow editing of all attributes in the KPI model, so this time we do not define any read-only attributes. Prior to generating the code we need to set the base package. We set this to be *org.eclipse.gmf.snapt*. Since the Ecore package is named "*kpi*" the model code will be generated into the *org.eclipse.gmf.snapt.kpi* bundle, adapters will be placed under the *org.eclipse.gmf.snapt.kpi.edit* bundle and the *org.eclipse.gmf.snapt.kpi.editor* bundle will contain the basic editor code. After the code is generated, a few modifications are required to remove the open / save functionality, and tune the KPI editor to smoothly cooperate with SNAPT.

6.6.2. Assigning KPOs

Once the KPI editor is developed, we still a way to assign KPOs to Services using the KPI Library. To do this we have implemented a new SNAPT view. A screenshot of this view is shown in figure 21. When a new view is added to an RCP application, several problems arise. The view needs to be in sync with the editors and other views of the application, so that the system is consistent. For example, when activating the KPOs view, the Properties view should not attempt to show properties of possible elements within the KPOs view, but continue displaying what it was showing before the KPOs view was activated. At the same time, the KPOs view must be aware of the active editor's selection, so that it can allow KPO assignments each time a Service is selected (recall that KPOs must be assigned to Services). This kind functionality is achieved using the *SelectionProviders* and *SelectionListeners* mechanism. Classes such as the SN and SSN

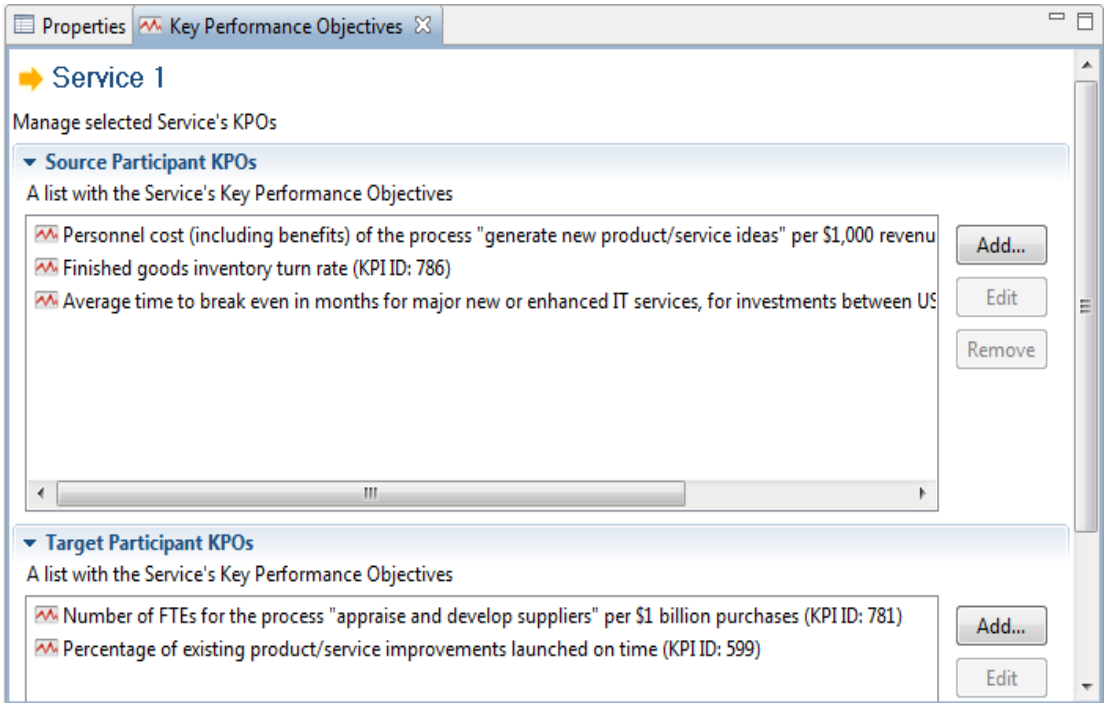

Figure 21. The KPOs view

editors, implement the SelectionProvider interface so that they can broadcast information on their selection. Classes interested in receiving this information should implement the *SelectionListener* interface and register themselves to receive selection information. Therefore, our KPOs view is a SelectionListener to SN and SSN editor selections, while being a SelectionProvider for other components to know how to behave when the KPOs view is selected. Using the KPOs view, SNAPT users can assign KPOs to both source and target Business Entities. To enable this, UI components that allow selection of KPIs from the KPI Library and editing of their targeted values have been developed. These components are called whenever a user selects to add or edit a

KPO through the KPOs view interface. Technically, the KPO UI components are standard Eclipse dialogs implementing the *FormDialog* class and having access to the underlying KPI Library model.

7. Discussion & Future Work

In this thesis we have proposed a Service Network model, we discussed an application of Service Network analysis to enhance service selection and presented the Service Network Analysis & Prediction Tool (SNAPT). Our work aims to serve as a basis for a universal Service Network-centric methodology to allow design, analysis and implementation of Services within complex Service Ecosystems. In the following sections we discuss what has already been done in this direction and what are the steps to follow in all aspects of the Service Network methodology.

7.1. Service Network analysis

In the past two decades, Service Networks have been studied by many researchers, as they comprise a convenient model to understand new business models that originate from the transformation of traditional vertical business hierarchies to modular groups of service deliveries. On the other hand, during the last couple of years, Service Network approaches have been adapted by IT managers and developers to analyze modern, complex Service-oriented software systems. In this work we have proposed a core Service Network model in an attempt to bridge these two otherwise incompatible worlds. However, our Service Network model could be further expanded to facilitate the needs of different stakeholders and experts.

Service Level Agreements (SLAs), are used to define service contracts between a service provider and a service consumer, to formally describe delivery times, performance and quality characteristics of a service [83]. SLAs are usually supervised by third-party authority entities. Although not part of the Service Network core model, SLAs and authority entities are closely related to the Service Network concept. The ways in which SLAs could be reflected to the Service Network level, and the roles that authorities can play in the network are an interesting open problem. Moreover, since KPOs are currently used to describe service-level strategic goals, a possible use of KPOs to semi-automatically inject QoS and performance characteristics to an SLA is to be investigated. As a result of this research, an extended SLA-oriented Service Network meta-model could be designed to allow business analysts to study, design and monitor SLAs.

The simple cost - revenue model proposed in this work to facilitate Service Network calculations, targets all different stakeholders participating in the Service design and analysis process. However, traditional Service Network analysis deals with network value [44][2] and targets the most economists and business analysts. In [84] an example of classic Service Network analysis and a methodology for simulating and predicting a Service Network's evolution is presented. To facilitate analysis and prediction of a Service Network's behavior in this context, a possible extension of the core Service Network model could also be examined. Verna Allee has conducted a considerable amount of research in the field of Service Value Networks and has developed a methodology to perform qualitative-oriented analysis to Service Networks. Although not discussed in this thesis, the Transformation Services Laboratory (TSL)[6] has defined a set of transformation rules in an attempt to converge our Service Network model with Allee's Service Value Network model; the results of this effort are yet to be presented in future work.

Since Service Networks can be seen as an abstraction of complex Service-Oriented Architecture (SOA) systems, a methodology to semi-automatically transform Service Network models to abstract business process models is needed. Steps towards this direction have also been conducted and an extended Service Network model was designed to facilitate this task; these are also planned to be presented in a future work. However, a dedicated methodology to allow design and implementation of fully-functional SOA systems that use Service Networks as part of the design as well as monitoring processes while enabling dynamic transformations of the underlying business model based on Service Network monitoring, is still to be discovered and defined. KPOs are of key importance in such a process. KPOs can be exploited to form the basis of KPI models used to monitor already running processes and vice-versa: Already measured KPIs should be compared to the KPOs found on the Service Network level and drive its transformation to improve performance.

7.2. Enhancing service selection based on Service Networks Analysis

As part of this thesis we contributed a process model for KPI-driven service discovery to enhance Service Network analysis and service selection. Recall that this effort applies to software-oriented service systems; ways of adopting this methodology in human-oriented ecosystems are yet to be discovered. The research draws on previous research in requirements-based and ontology-driven service discovery, as well as on established Service Network analysis techniques. In particular, the process model is supported using

[6] http://www.tsl.gr/

software components that (1) discover candidate services based on quality of service characteristics and use them to dynamically create new Service Networks within a service ecosystem, and (2) exploit newly created Service Networks to improve the service discovery and selection process. Using consumer-related information such as customer demands and feedback we are able to monitor and select services more effectively. An increase or decrease of customer satisfaction will encourage service providers to react to user feedback, thus enabling a more competitive market. As this work reports the early results from ongoing work into the use of requirements-based service discovery in Service Network analysis and service ecosystem modeling, the report is limited to describing the approach and demonstrating it using an example.

The next stages in this direction are to evaluate the utility of the process model and to complete the implementation of the first version of the KPI-driven service discovery engine. During evaluation it is to be explored whether the presented approach can assist end-user developers such as automotive engineers to enhance customer satisfaction by applying ACSI to the service discovery process.

In order to increase the overall customer satisfaction index, the impact of *user models* on the emergence of new Service Networks and on the enhancement of the service ecosystem should be investigated. User modeling is an essential technique that can support the personalization of services for adaptation to a user. In real-world situations, it is unlikely that every service customer has the same needs and shares the same expectations for service quality. In our automotive example of section 4.4, let us assume two user models for a service consumer – a car driver who uses his car for business (*professional*) and a car driver that uses the car for leisure (*tourist*). The professional on the one hand is not concerned about costs as long as the navigation service satisfies his business needs. On the other hand, the tourist is more concerned on keeping the costs low since her budget might be limited. Therefore, service consumers that belong to the professional user model will end up preferring different service providers than service consumers that belong to the tourist user model. Extending the process to consider user models will ensure that different classes of users will be able to select candidate services and service providers that facilitate their personal characteristics or preferences. The exact ways that user profile and models will affect service discovery and selection as well as their impact on Service Networks is work to be done.

7.3. SNAPT functionality & future aspects

The Service Network Analysis & Prediction Tool (SNAPT) is a Model-Driven Architecture (MDA) platform that enables modeling and analysis of Service Networks. Since Service

Network theory is constantly evolving, SNAPT was developed in a modular way, to host future task-specific, extended Service Network models. SNAPT incorporates the Service Network and KPI models discussed in this thesis, facilitates cost - revenue analysis and includes an extensive feature set to achieve its partial goals.

As discussed in previous section, the core Service Network model could be expanded in many ways to facilitate both analysis and implementation of complex Services. SNAPT should be in the center of these emerging methodologies. To facilitate value analysis and prediction, SNAPT could be connected to well-established simulation platforms such as iThink [56] or VenSim [55]. At the same time, it could also serve as a bridge to business process modeling platforms such as the IBM WebSphere [63] software family.

An additional future challenge for SNAPT, could be the employment of Roles that serve as network templates. Depending on the Role a Business Entity plays in a network, there might be a set of services that the Business Entity must offer and consume to fulfill its mission. One could study several Service Network examples and define a library of roles that would be used to effectively model Service Networks. A respective library of role template business processes could also be developed and mapped to the Service Network level to facilitate the transition to actual running processes.

Since SNAPT is a modular system, a way to get automatically updated with features, fixes and new components might be needed. A repository of SNAPT bundles should also be developed so that users can only install the components they use. For example, a service designer might need the Verna Allee or the Service Network simulation expansions. On the other hand, a IT manager could receive a complete Service Network model and use only business process expansions to eventually develop a SOA system. Eclipse already offers an update mechanism that is not by default included in RCP applications. The incorporation of this Eclipse mechanism in SNAPT could be a reasonable starting point to enable the install, uninstall and update of SNAPT bundles.

Bibliography

1. **A. Barros, M. Dumas and P. Bruza.** *The Move to Web Service Ecosystems.* s.l. : BPTrends, 2005.

2. **N. S. Caswell, C. Nikolaou, J. Sairamesh, M. Bitsaki, G. D. Koutras and G. Iacovidis.** *Estimating value in service systems: A case study of a repair service system.* 1, s.l. : IBM, 2008, IBM Systems Journal, Vol. 47.

3. **M. Papazoglou.** *Web Services: Principles and Technology.* s.l. : Prentice Hall, 2007.

4. **RosettaNet.** RosettaNet Home. [Online] July 2010. http://www.rosettanet.org/.

5. **OpenTravel Alliance (OTA).** OpenTravel. [Online] July 2010. http://www.opentravel.org/.

6. **B. Wetzstein, O. Danylevych, F. Leymann, M. Bitsaki, C. Nikolaou, W. Van den Heuvel and M. Papazoglou.** *Towards Monitoring of Key Performance Indicators Across Partners in Service Networks.*2008. International Workshop on Service Monitoring, Adaptation and Beyond.

7. **B. Wetzstein, D. Karastoyanova and F. Leymann.** *Towards Management of SLA-Aware Business Processes Based on Key Performance Indicators.* 2008. 9th Workshop on Business Process Modeling, Development, and Support (BPMDS'08).

8. **M. Porter.** *Competitive Advantage: Creating and Sustaining Superior Performance.* s.l. : Free Press, 1985.

9. **J. Peppard and A. Rylander.** *From Value Chain to Value Network: Insights for Mobile Operators.* 2-3, s.l. : Elsevier, 2006, Vol. European Management Journal.

10. **M. Bitsaki, O. Danylevych, W.J. van den Heuvel, G. Koutras, F. Leymann, M. Mancioppi, C. Nikolaou and M.P. Papazoglou.** *An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network.* 2008. ServiceWave 2008.

11. **V. Allee.** *Value Network Analysis and Value Conversion of Tangible and Intangible Assets.* 1, s.l. : Emerald Insights, 2008, Journal of Intellectual Capital, Vol. 9.

12. **J. Gordijn and H. Akkermans.** *Designing and Evaluating e-Business Models.* 4, s.l. : IEEE, 2001, Intelligent Systems, Vol. 16.

13. **S. L. Vargo and R. F. Lusch.** *From goods to service(s): Divergences and convergences of logics.* 3, 2008, Industrial Marketing Management, Vol. 37.

14. **J. Spohrer, P.P. Maglio, J Bailey and D. Gruhl.** *Steps Toward a Science of Service Systems.* 1, s.l. : IEEE Computer Society Press, 2007, Computer, Vol. 40.

15. **R. Karni and M. Kaner.** An engineering tool for the conceptual design of service systems. *Advances in Services Innovations.* s.l. : Springer Berlin Heidelberg, 2007.

16. **J. Spohrer, S. L. Vargo, N. Caswell and P. P. Maglio.** *The Service System Is the Basic Abstraction of Service Science.* 2008. 41st Annual Hawaii International Conference on System Sciences.

17. **G. Scheithauer, S. Augustin and G. Wirtz.** *Describing Services for Service Ecosystems.* s.l. : Springer Berlin, 2009, Lecture Notes in Computer Science, Vol. 5472/2009.

18. **G. Scheithauer.** *Business Service Description Methodology for Service Ecosystems.* 2009. CAiSE-DC'09 16th Doctoral Consortium held in conjunction with CAiSE'09 Conference Amsterdam. Vol. 479.

19. **A. Barros and M. Dumas.** *The Rise of Web Service Ecosytems.* 5, 2006, IT Professional , Vol. 8.

20. **S. Alter.** *Service System fundamentals: Work system, Value chain, and Life cycle.* 1, s.l. : IBM, 2008, Systems Journal, Vol. 47.

21. **V. Allee.** Value Network Mapping Basics. *Open Value Networks.* [Online] August 2010. http://www.openvaluenetworks.com/howToGuides/ValueNetworkMappingBasics.pdf.

22. **V. Allee**. *A Value Networks approach for Modeling and Measuring Intangibles, White Paper.* 2002.

23. **V. Allee**. Value Network Insights. *Value Networks.* [Online] August 2010. www.valuenetworks.com.

24. **J. Gordijn and H. Akkermans.** *Value-based requirements engineering: Exploring innovative e-commerce ideas.* 2, s.l. : Springer London, 2003, Requirements Engineering Journal, Vol. 8.

25. **J. Gordijn, P. van Eck and R. Wieringa.** *Requirements Engineering Techniques for e-Services.* s.l. : MIT press, 2008, Readings in Service-Oriented Computing: the Web-Services Phenomenon.

26. **J. Gordijn and H. Akkermans.** *Ontology-based operators for e-business model de- and reconstruction.* s.l. : ACM, 2001. 1st international conference on Knowledge capture.

27. **e3value.** Tools for e3value. [Online] http://www.e3value.com/tools/.

28. **V. Kartseva, J. Gordijn and Yao-Hua Tan.** *Designing Value-based Inter-organizational Controls Using Patterns.* s.l. : Springer Berlin, 2009, Lecture Notes in Business Information Processing, Vol. 14.

29. **H. Weigand, P. Johannesson, B. Andersson, M. Bergholtz, A. Edirisuriya and T. Llayperuma.** *Strategic Analysis Using Value Modeling--The c3-Value Approach.* s.l. : IEEE Computer Society, 2007. 40th Annual Hawaii International Conference on System Sciences.

30. **G. D. Koutras.** *Model transformations to Leverage service networks and implementation of value network tool.* Computer Science Department. s.l. : University of Crete, 2009.

31. **B. Blau, J. Krämer, T. Conte and C. van Dinther.** *Service Value Networks.* s.l. : IEEE, 2009. Conference on Commerce and Enterprise Computing.

32. **B. Blau, C. van Dinther, T. Conte, Yongchun Xu and C. Weinhardt.** *How to Coordinate Value Generation in Service Networks - A Mechanism Design Approach.* 5, s.l. : Gabler Verlag, 2009, Business & Information Systems Engineering, Vol. 1.

33. **C. van Dinther, B. Blau and T. Conte.** *Strategic Behavior in Service Networks under Price and Service Level Competition.* 2009. 9th International Conference on Business Informatics .

34. **C. T. Fitz-Gibbon.** *Performance Indicators.* s.l. : Multilingual Matters Limited, 1990.

35. **R. Simmons.** *Performance Measurement and Control Systems for Implementing Strategy Text and Cases.* s.l. : Prentice Hall, 1999.

36. **V. Kellen.** Business Performance Measurement. [Online] 2003. http://www.kellen.net/bpm.htm.

37. **D. Parmenter.** *Key Performance Indicators (KPI): Developing, Implementing,and Using Winning KPIs.* s.l. : Wiley, 2007.

38. **R. Shandi.** *Key Performance Indicators – Measuring Performance in Oil & Gas EPC Industry.* s.l. : Delft University of Technology, 2009.

39. **EFFAS and DFVA.** *KPIs for ESG – A guideline for the Integration of ESG into Financial Analysis and Corporate Valuation, White Paper.* 2010.

40. **Mirror42.** KPI Library. [Online] August 2010. http://www.kpilibrary.com/.

41. **APQC.** Process Classification Framework. [Online] August 2010. http://www.apqc.org/pcf.

42. **R. C. Basole and W. B. Rouse.** *Complexity of Service Value Networks: Conceptualization and Empirical Investigation.* 1, s.l. : IBM, 2008, Systems Journal, Vol. 47.

43. **M. Bitsaki, N. Caswell, G. Iacovidis, C. Nikolaou, J. Sairamesh and S. Tai.** *Estimating Value in Value Networks - A Case Study from Pharmaceutical Industry.* 2007. 16th Annual Frontiers in Service Conference.

44. **V. Allee.** *Value Network Case Study, White Paper.* 2006.

45. **S. van Middendorp.** *The three value networks in concert - A case study of Mindz and Seats2meet.* s.l. : Miles Ahead Business Jazz BV, 2010.

46. **C. Fornell, M. D. Johnson, E. W. Anderson, J. Cha and B. E. Bryant.** *The American Customer Satisfaction Index: Nature, Purpose and Findings.* 1996, Journal of Marketing, Vol. 60.

47. **ACSI.** American Customer Satisfaction Index. [Online] August 2010. http://www.theacsi.org/.

48. **K. Zachos, N.A.M Maiden, S.Jones and X.Zhu.** *Discovering Web Services To Specify More Complete System Requirements.* 2007. 19th Conference on Advanced Informa-tion System Engineering (CAiSE'07). pp. 142 - 157.

49. **J. Walkerdine, J., Hutchinson, P. Sawyer, G. Dobson, V. Onditi.** *A Faceted Approach to Service Specification.* Mauritus : s.n., 2007. 2nd Int. Conf. on Internet and Web Applications and Services.

50. **G. Dobson, P. Sawyer.** *Revisiting Ontology-Based Requirements Engineering in the Age of the Semantic Web.* Norway : s.n., 2006. International seminar on Dependable Requirements Engineering of Computerised Systems.

51. **J. Bloomburg.** Competitive SOA. *ZapThink.* [Online] February 2007. http://www.zapthink.com/2007/02/22/competitive-soa/.

52. **K. Zachos, G. Dobson and P. Sawyer.** *Ontology-aided Translation in the Comparison of Candidate Service Quality.* Barchelona : s.n., 2008. Service-Oriented Computing: Consequences for Engineering Requirements (SOCCER 08).

53. **G. Dobson, R. Lock and I. Sommerville.** *Quality of Service Requirements Specification using an Ontology.* 2005. Service Oriented Computing: Consequences for Engineering Requirements (SOCCER 05).

54. **Service Centric System Engineering (SeCSE).** *EU 511680 Integrated Project.* [Online] August 2010. http://www.secse-project.eu/.

55. **Ventana Systems.** VENSIM. [Online] August 2010. http://www.vensim.com/.

56. **iSee Systems.** iThink - Systems Thinking for Business. [Online] August 2010. http://www.iseesystems.com/softwares/Business/ithinkSoftware.aspx.

57. **OMG.** Documents Associated with CORBA, 3.1. [Online] January 2008. http://www.omg.org/spec/CORBA/3.1/.

58. **OMG**. Unified modeling Language. [Online] August 2010. http://www.uml.org/.

59. **OMG**. MOF 2.0 / XMI Mapping Specification, v2.1.1. [Online] August 2010. http://www.omg.org/technology/documents/formal/xmi.htm.

60. **A. Brown.** An introduction to Model Driven Architecture. [Online] IBM, Feb 2004. http://www.ibm.com/developerworks/rational/library/3100.html.

61. **M1 Global Solutions.** MDA Success Story - Model Driven Software Development and Offshore Outsourcing. [Online] OMG, August 2010. http://www.omg.org/mda/mda_files/M1Global.htm.

62. **S. E. Slack.** The business analyst in model-driven architecture. [Online] IBM, 2008 September. http://www.ibm.com/developerworks/library/ar-bamda/index.html.

63. **IBM.** IBM Software - Websphere. [Online] August 2010. http://www-01.ibm.com/software/websphere/.

64. **Microsoft.** Microsoft .NET. [Online] August 2010. http://www.microsoft.com/net/.

65. **A. P. Singh and A. Jain.** Model-Driven Architecture with GMF. [Online] December 2009. http://www.devx.com/architect/Article/43366/1763/.

66. **Eclipse Foundation.** Eclipse Modeling Framework Project (EMF). [Online] August 2010. http://www.eclipse.org/modeling/emf/.

67. **Eclipse Foundation**. Graphical Editing Framework (GEF). [Online] August 2010. http://www.eclipse.org/gef/.

68. **R. Gronback and F. Plante.** *Introduction to the Eclipse Graphical Modeling Framework.* s.l. : Eclipse Foundation, 2006. EclipseCon 06.

69. **L. Vogel.** Eclipse Extension Points and Extensions - Tutorial. [Online] June 2010. http://www.vogella.de/articles/EclipseExtensionPoint/article.html.

70. **A. Tikhomirov and A. Shatalin.** *Introduction to the Graphical Modeling Framework – Network domain demo.* s.l. : Eclipse Foundation, 2008. EclipseCon 08.

71. **H. Jorgensen.** Is Eclipse GMF a Suitable Platform for Model Driven Applications? [Online] March 2010. http://activeknowledgemodeling.com/2010/03/16/is-eclipse-gmf-a-suitable-platform-for-model-driven-applications/.

72. **J. Koehnlein.** GMF Tools - A collection of tools and extensions for the Graphical Modeling Framework. [Online] August 2010. http://code.google.com/p/gmftools/.

73. **Eclipse Foundation.** Eclipse. [Online] August 2010. http://www.eclipse.org/.

74. **Eclipse Foundation**. Rich Client Platform. [Online] August 2010. http://wiki.eclipse.org/index.php/Rich_Client_Platform.

75. **O. Gruber, B. J. Hargrave, J. McAffer, P. Rapicault and T. Watson.** *The Eclipse 3.0 platform: adopting OSGi technology.* 2, s.l. : IBM, 2005, Systems Journal, Vol. 44.

76. **OSGi Alliance.** About the OSGi Service Platform. [Online] July 2004. http://www.osgi.org/documents/osgi_technology/osgi-sp-overview.pdf.

77. **Eclipse Foundation.** SWT: The Standard Widget Toolkit. [Online] August 2010. http://www.eclipse.org/swt/.

78. **Eclipse Foundation**. JFace. [Online] August 2010. http://wiki.eclipse.org/index.php/JFace.

79. **D. Gallardo.** Getting Started with the Eclipse Workbench. [Online] May 2003. http://www.devx.com/opensource/Article/15779/0/.

80. **OMG.** Documents associated with Object Constraint Language, Version 2.2. [Online] Febrary 2010. http://www.omg.org/spec/OCL/2.2/.

81. **J. Koehnlein.** Shared Editing Domain. [Online] October 2009. http://code.google.com/p/gmftools/wiki/SharedEditingDomain.

82. **Apache Foundation.** Apache POI - the Java API for Microsoft Documents. [Online] August 2010. http://poi.apache.org/.

83. **A. N. Hiles.** *Service Level Agreements: Panacea or Pain?* 2, s.l. : MCB UP, 1994, The TQM Magazine, Vol. 6.

84. **M. Voskakis.** *Simulation and optimization of value in Service Networks.* Computer Science Department. s.l. : University of Crete, 2010.

Appendix: The Ecore Domain Models

The complete Ecore model files –expressed in Ecore XML - for SN and SSN editors (SNE model – see section 6.1.1) as well as the KPI editor (see section 6.6.1) are shown below.

1. Service Network Ecore Domain Model

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="sn"
         nsURI="http://tsl.gr/snapt/sn" nsPrefix="">
   <eClassifiers xsi:type="ecore:EClass" name="ServiceNetwork">
      <eStructuralFeatures xsi:type="ecore:EReference" name="Offerings"
            lowerBound="1" upperBound="-1" eType="#//Service"
            containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference"
            name="BusinessEntities" lowerBound="2" upperBound="-1"
            eType="#//BusinessEntity" containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Enablements"
            upperBound="-1" eType="#//ServiceEnablement"
            containment="true"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute"
            name="ServiceNetworkId" unique="false" lowerBound="1"
            eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt"
            defaultValueLiteral="1" unsettable="true"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="BusinessEntity"
         abstract="true">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="TotalCost"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="TotalRevenue"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Role"
            unique="false" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Participant"
            eSuperTypes="#//BusinessEntity">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="ParticipantId"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt"
            unsettable="true"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Enabler"
         eSuperTypes="#//BusinessEntity">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="EnablerId"
```

```
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt"
            unsettable="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="EndCustomer"
          eSuperTypes="#//BusinessEntity">
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="EndCustomerId"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt"
            unsettable="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="ServiceSubNetwork"
          eSuperTypes="#//BusinessEntity #//ServiceNetwork">
       <eStructuralFeatures xsi:type="ecore:EReference"
            name="IncomingServices" upperBound="-1" eType="#//SSNInput"
            containment="true"/>
       <eStructuralFeatures xsi:type="ecore:EReference"
            name="OutgoingServices" upperBound="-1" eType="#//SSNOutput"
            containment="true"/>
       <eStructuralFeatures xsi:type="ecore:EReference" name="IncomingPorts"
            upperBound="-1" eType="#//SSNInputPort" containment="true"/>
       <eStructuralFeatures xsi:type="ecore:EReference" name="OutgoingPorts"
            upperBound="-1" eType="#//SSNOutputPort" containment="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="Service">
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="Revenue"
            unique="false" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
       <eStructuralFeatures xsi:type="ecore:EReference"
            name="SourceBusinessEntity" lowerBound="1"
            eType="#//BusinessEntity"/>
       <eStructuralFeatures xsi:type="ecore:EReference"
            name="TargetBusinessEntity" lowerBound="1"
            eType="#//BusinessEntity"/>
       <eStructuralFeatures xsi:type="ecore:EReference" name="SourceKPOs"
            upperBound="-1" eType="#//KPO" containment="true"/>
       <eStructuralFeatures xsi:type="ecore:EReference" name="TargetKPOs"
            upperBound="-1" eType="#//KPO" containment="true"/>
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="ServiceId"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt" unsettable="true"
            iD="true"/>
    </eClassifiers>
    <eClassifiers xsi:type="ecore:EClass" name="KPO">
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="KPOId"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EInt"
            unsettable="true"/>
       <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
```

```
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Type"
          eType="#//KPOType"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Value"
          lowerBound="1" eType="ecore:EDataType
          http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="KPOType">
    <eLiterals name="Unspecified"/>
    <eLiterals name="Number"/>
    <eLiterals name="Duration"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ServiceEnablement">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
          unique="false" lowerBound="1" eType="ecore:EDataType
          http://www.eclipse.org/emf/2002/Ecore#//EString"
          unsettable="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Revenue"
          unique="false" eType="ecore:EDataType
          http://www.eclipse.org/emf/2002/Ecore#//EDouble"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute"
          name="ServiceEnablementId" unique="false" lowerBound="1"
          eType="ecore:EDataType
          http://www.eclipse.org/emf/2002/Ecore#//EInt" unsettable="true"
          iD="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="SourceEnabler"
          lowerBound="1" eType="#//Enabler"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="TargetService"
          lowerBound="1" eType="#//Service"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SSNInput">
    <eStructuralFeatures xsi:type="ecore:EReference"
          name="TargetBusinessEntity" lowerBound="1"
          eType="#//BusinessEntity"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="IncomingPort"
          lowerBound="1" eType="#//SSNInputPort"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SSNOutput">
    <eStructuralFeatures xsi:type="ecore:EReference"
          name="SourceBusinessEntity" lowerBound="1"
          eType="#//BusinessEntity"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="OutgoingPort"
          lowerBound="1" eType="#//SSNOutputPort"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SSNInputPort">
    <eStructuralFeatures xsi:type="ecore:EReference"
          name="IncomingService" lowerBound="1" eType="#//Service"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
          unique="false" lowerBound="1" eType="ecore:EDataType
          http://www.eclipse.org/emf/2002/Ecore#//EString"
          unsettable="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="SSNOutputPort">
    <eStructuralFeatures xsi:type="ecore:EReference"
          name="OutgoingService" lowerBound="1" eType="#//Service"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
          unique="false" lowerBound="1" eType="ecore:EDataType
```

```
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
   </eClassifiers>
</ecore:EPackage>
```

2. KPI Ecore Model

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi=http://www.omg.org/XMI
      xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
      xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="kpi"
      nsURI="http://tsl.gr/snapt/kpi" nsPrefix="">
   <eClassifiers xsi:type="ecore:EClass" name="KPILibrary">
      <eStructuralFeatures xsi:type="ecore:EReference" name="Categories"
            upperBound="-1" eType="#//Category" containment="true"
            eKeys="#//Category/Id"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="KPI">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Id"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            defaultValueLiteral="" unsettable="true"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Description"
            unique="false" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Category">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Id"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="Subcategories"
            upperBound="-1" eType="#//Subcategory" containment="true"
            eKeys="#//Subcategory/Id"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
   </eClassifiers>
   <eClassifiers xsi:type="ecore:EClass" name="Subcategory">
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Id"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"
            unsettable="true"/>
      <eStructuralFeatures xsi:type="ecore:EAttribute" name="Name"
            unique="false" lowerBound="1" eType="ecore:EDataType
            http://www.eclipse.org/emf/2002/Ecore#//EString"/>
      <eStructuralFeatures xsi:type="ecore:EReference" name="KPIs"
            upperBound="-1" eType="#//KPI" containment="true"
            eKeys="#//KPI/Id"/>
   </eClassifiers>
</ecore:EPackage>
```